ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Văn Định

ĐÁNH GIÁ HIỆU NĂNG MỘT SỐ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU NOSQL

LUẬN VĂN THẠC SĨ KHOA HỌC MÁY TÍNH

HÀ NỘI - 2020

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Văn Định

ĐÁNH GIÁ HIỆU NĂNG MỘT SỐ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU NOSQL

Ngành: Công Nghệ Thông Tin
Chuyên ngành: Khoa Học Máy Tính
Mã số: 8480101.01

LUẬN VĂN THẠC SĨ KHOA HỌC MÁY TÍNH

NGƯỜI HƯỚNG DẪN KHOA HỌC: 1. PGS.TS. Nguyễn Hoài Sơn
2. TS. Phạm Mạnh Linh

HÀ NỘI - 2020

LỜI CAM ĐOAN

Những nội dung trình bày trong luận văn là những kiến thức của riêng cá nhân tôi tích lũy trong quá trình học tập, nghiên cứu, không sao chép lại một công trình nghiên cứu hay luận văn của bất cứ tác giả nào khác.

Trong nội dung của nội dung của luận văn, những phần tôi nghiên cứu, trích dẫn đều được nêu trong phần các tài liệu tham khảo, có nguồn gốc, xuất xứ, tên tuổi của các tác giả, nhà xuất bản rõ ràng.

Những điều tôi cam kết hoàn toàn là sự thật, nếu sai, tôi xin chịu mọi hình thức xử lý kỷ luật theo quy định.

Hà Nội, Ngày tháng 9 năm 2020

Tác giả luận văn

Nguyễn Văn Định

LỜI CẢM ƠN

Em xin bày tỏ lòng biết ơn chân thành và sâu sắc đến PGS.TS Nguyễn Hoài Sơn và TS Phạm Mạnh Linh, khoa Công nghệ thông tin - Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội, các thầy đã dành nhiều thời gian tận tình chỉ bảo, hướng dẫn em trong suốt quá trình tìm hiểu, triển khai và nghiên cứu đề tài. Hai thầy là người đã định hướng và đưa ra nhiều góp ý quý báu trong quá trình em thực hiện luận văn này.

Em xin chân thành cảm ơn chân thành tới toàn thể các thầy giáo, cô giáo trong khoa Công nghệ thông tin - Trường Đại học Công nghệ Hà Nội - Đại học Quốc gia Hà Nội đã dạy bảo tận tình, trang bị cho em những kiến thức quý báu, bổ ích và tạo điều kiện thuận lợi trong suốt quá trình em học tập và nghiên cứu tại trường.

Do có nhiều hạn chế về thời gian và kiến thức nên luận văn không tránh khỏi những thiếu sót, rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô và các bạn cùng quan tâm.

Luận văn này được tài trợ bởi Học viện Khoa học và Công nghệ và Viện Công nghệ thông tin, Viện hàn lâm Khoa học và Công nghệ Việt Nam từ đề tài mã số GUST.STS.ĐT2019-TT02.

Cuối cùng em xin gửi lời chúc sức khỏe và thành đạt tới tất cả quý thầy cô, quý đồng nghiệp cùng toàn thể gia đình và bạn bè.

Xin chân thành cảm ơn!

DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT

ACID	Atomicity, Consistency, Isolation, and durability.
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
BLOB	Binary large object
BSON	Binary JSON
CAP	Consistency, Availability and Partition Tolerance.
CPU	Platform as a Service
CQL	Cassandra Query Language
CRUD	Create, Read, Update, Delete
EBS	Elastic Block
EC2	Elastic Compute Cloud
FTP	File Transfer Protocol
HDFS	Hadoop Distributed File System
IBM	International Business Machines
IoMT	Internet of Medical Things
IoT	Internet of Things

JSON	JavaScript Object Notation
NIST	National Institute of Standards and Technology
OLTP	On-line Transactional Processing
OLTP	On-line transactional processing
RDBMS	Relational Database Management System
Rhino DHT	Lưu trữ liên tục & phân tán
RM	Readmodify-write
RPC	Remote Procedure Calls
TPC	Transaction Processing Performance Council
TPS	transaction per second
URI	Uniform Resource Identifier
VM	Virtual machine
XML	eXtensible Markup Language
YAML	YAML Ain't Markup Language
YCSB	Yahoo Cloud Serving Benchmark

DANH SÁCH CÁC BẢNG

DANH SÁCH CÁC HÌNH VẼ

MỤC LỤC

LỜI MỞ ĐẦU

Hệ quản trị cơ sở dữ liệu quan hệ truyền thống của những năm 1970 được thiết kế để phù hợp với các yêu cầu của các ứng dụng xử lý giao dịch trực tuyến (OLTP) như là các giải pháp "một kích thước phù hợp với tất cả"[13]. Các hệ thống này thường được lưu trữ trên một máy chủ duy nhất, nơi quản trị cơ sở dữ liệu đáp ứng với sự tăng trưởng dữ liệu bằng cách tăng tốc độc xử lý của CPU, dung lượng bộ nhớ và tốc độ của đĩa cứng trên máy chủ duy nhất đó, tức là bằng cách mở rộng theo chiều dọc. Hệ quản trị cơ sở dữ liệu quan hệ vẫn còn phù hợp trong môi trường máy tính hiện đại ngày nay, tuy nhiên, những hạn chế của khả năng mở rộng theo chiều dọc là mối quan tâm lớn nhất.

Khối lượng dữ liệu được sử dụng bởi nhiều tổ chức trong những năm gần đây đã lớn hơn dung lượng của một máy chủ duy nhất, do sự bùng nổ của web. Do đó, các công nghệ và kỹ thuật mới đã được phát triển để giải quyết những vấn đề của dữ liệu. Năm 2010, người ta tuyên bố rằng Facebook đã lưu trữ kho dữ liệu lớn nhất thế giới được lưu trữ trên HDFS (hệ thống phân tán của Hadoop), bao gồm 2000 máy chủ, tiêu thụ 21 Petabyte (PB) dung lượng lưu trữ [15], nhanh chóng tăng lên 100 PB vào đầu năm 2012 [3].

Tình hình này còn phức tạp hơn bởi thực tế các ứng dụng OLTP truyền thống chỉ là một tập hợp con của các trường hợp sử dụng mà các công nghệ mới này phải tạo điều kiện thuận lợi bởi một loạt các yêu cầu ứng dụng hiện đại của ngành công nghiệp [10]. Do đó, các công nghệ dữ liệu lớn được yêu cầu mở rộng quy mô theo một cách khác để khắc phục các hạn chế về tốc độ xử lý, dung lượng bộ nhớ và tốc độ ghi của đĩa. Khả năng mở rộng theo chiều ngang đã trở thành trọng tâm mới cho các hệ thống lưu trữ dữ liệu; cung cấp khả năng truyền dữ liệu từ nhiều máy chủ.

Độ co giãn là một thuộc tính của khả năng mở rộng theo chiều ngang, cho phép tăng thông lượng tuyến tính khi nhiều máy được thêm vào một cụm. Môi trường điện toán hiện đại cũng chứng kiến sự gia tăng số lượng các nhà cung cấp dịch vụ điện toán đám mây. Nhiều trong số đó vốn có tính đàn hồi, ví dụ như AWS của Amazon [2] và Rackspace [14], cung cấp khả năng mở rộng theo chiều ngang một cách dễ dàng và với chi phí thấp cho người dùng.

Nhiều kho dữ liệu mới đã được thiết kế với suy nghĩ về sự thay đổi cảnh quan này, một số trong số đó thậm chí còn được thiết kế để hoạt động độc lập

trên điện toán đám mây, ví dụ PNUTS của Yahoo, BigTable của Google và Amazon của DynamoDB. Các hệ thống lưu trữ dữ liệu mới này thường được gọi là kho dữ liệu NoSQL, viết tắt của "Not Only SQL"; vì chúng không sử dụng ngôn ngữ truy vấn có cấu trúc (SQL) hoặc có mô hình quan hệ. Hewitt [9] đề xuất rằng thuật ngữ này cũng có nghĩa là các hệ thống truyền thống không nên là lựa chọn duy nhất để lưu trữ dữ liệu.

Hiện nay, có rất nhiều giải pháp lưu trữ dữ liệu NoSQL và một số công ty sử dụng các giải pháp này để giải quyết với những thách thức về khả năng mở rộng dữ liệu và chọn giải pháp tốt nhất cho trường hợp sử dụng cụ thể của mình. Dữ liệu mà các công ty thu thập có giá trị cao và việc truy vấn dữ liệu này cần có tính sẵn sàng cao [15]. Nhu cầu về tính khả dụng cao của dữ liệu trở nên đặc biệt rõ ràng trong các ứng dụng Web mà bản thân chúng có thể đã quen với việc tương tác hàng ngày, ví dụ như các nền tảng truyền thông xã hội như Facebook và Twitter.

Một tính năng của khả năng mở rộng theo chiều ngang là thêm các máy chủ tạo thành một cụm. Số lượng máy chủ lớn cũng làm tăng tần suất các nút gặp sự cố. Cơ chế chính mà dữ liệu NoSQL lưu trữ ở mức độ sẵn sàng cao để khắc phục tỷ lệ lỗi cao và đáp ứng kỳ vọng thông qua việc mở rộng. Việc sao chép không chỉ mang lại khả năng dự phòng cao hơn trong trường hợp hỏng hóc mà còn có thể giúp tránh mất dữ liệu (bằng cách khôi phục dữ liệu bị mất từ một bản sao) và cải thiện hiệu suất (bằng cách dàn trải tải trên nhiều bản sao) [4].

Như đã đề cập về nhiều tính năng và lợi thế của việc sử dụng một số hệ quản trị cơ sở dữ liệu NoSQL nhưng điều đó không có nghĩa là nó phù hợp với mọi loại ứng dụng. NoSQL cũng có một số nhược điểm, vì vậy việc lựa chọn một số hệ quản trị cơ sở dữ liệu phù hợp tùy thuộc vào loại ứng dụng của chúng ta hoặc bản chất của dữ liệu. Một số dự án phù hợp hơn với việc sử dụng hệ quản trị cơ sở dữ liệu SQL, trong khi những dự án khác hoạt động tốt với NoSQL. Một số có thể sử dụng thay thế cả hai. Nếu ứng dụng đang lưu trữ dữ liệu giao dịch thì SQL là lựa chọn tốt nhất. Hệ quản trị cơ sở dữ liệu quan hệ được tạo ra để xử lý các giao dịch và chúng hoạt động rất tốt với điều đó. Nếu dữ liệu là phân tích thì hãy nghĩ về NoSQL, SQL không bao giờ dành cho phân tích dữ liệu và với một lượng lớn dữ liệu có thể trở thành một vấn đề.

Trong luận văn này, tôi sẽ đánh giá về hiệu suất của bốn hệ quản trị cơ sở dữ liệu NoSQL khác nhau như MongoDB, Cassandra, Redis và OrientDB bằng

cách chạy một loạt các kịch bản được tùy chỉnh để mô hình hóa các khối lượng truy vấn CURD trong thế giới thực khác nhau. Tôi so sánh hiệu suất của các hệ quản trị cơ sở dữ liệu NoSQL này liên quan đến hiệu suất truy vấn, dựa trên kịch bản đọc, chèn, quét và cập nhật bằng cách chạy các kịch bản bằng công cụ Yahoo! Cloud Serving Benchmark (YCSB) và đưa ra nhận xét về bốn hệ quản trị cơ sở dữ liệu này.

Chương 1: Dữ liệu lớn và các công cụ đánh giá hiêu năng

Chương này cung cấp kiến thức tổng quan và nền tảng về dữ liệu lớn và mối quan hệ giữa chúng. Giới thiệu các công cụ đánh giá hiệu năng cho dữ liệu lớn phổ biến hiện nay và tìm hiểu về các thuộc tính của từng loại.

1.1 Dữ liệu lớn

Mạng xã hội, phân tích web, thương mại điện tử thông minh thường cần quản lý dữ liệu ở quy mô quá lớn đối với hệ quản trị cơ sở dữ liệu quan hệ truyền thống (RDBMS). Mặc dù các nhà cung cấp hệ quản trị cơ sở dữ liệu đã cố gắng xử lý và lưu trữ khối lượng lớn dữ liệu với hiệu suất tốt hơn nhưng trong thập kỷ qua, sự gia tăng đáng kể về kích thước dữ liệu đã gây ra một số vấn đề để xử lý và lưu trữ dữ liệu với khả năng truy cập nhanh hơn và chi phí lưu trữ thấp hơn. Hệ quản trị cơ sở dữ liệu NoSQL được coi là một thành phần quan trọng của dữ liệu lớn khi truy xuất và lưu trữ một lượng lớn dữ liệu với chi phí lưu trữ giảm [10]. Ví dụ: theo báo cáo, chi phí lưu trữ RDBMS truyền thống trung bình là $30.000+ cho mỗi terabyte, trong khi lưu trữ cho hệ quản trị cơ sở dữ liệu NoSQL trung bình $1000 mỗi terabyte.

Từ điển tiếng Anh Oxford định nghĩa dữ liệu lớn như sau [19]:

"*Các tập dữ liệu cực lớn có thể được phân tích tính toán để tiết lộ các mẫu, xu hướng và liên kết, đặc biệt liên quan đến hành vi và tương tác của con người*"

Dữ liệu lớn đang đóng một vai trò quan trọng để cải thiện xã hội hiện đại. Mọi người có thể sử dụng các công nghệ và công cụ mới để cải thiện mọi khía cạnh của cuộc sống, bao gồm sức khỏe, chăm sóc y tế, để tăng tốc khám phá và đổi mới. Bên cạnh sự cải tiến và những khía cạnh tích cực này, Dữ liệu lớn đang đặt ra nhiều thách thức khác nhau cho cả Chính phủ và Người dân vì những công nghệ dữ liệu này đang trở nên rất phổ biến và khó hiểu. Ngoài ra, có nguy cơ lạm dụng và lạm dụng các bộ dữ liệu lớn này. Vì vậy, có nhiều vấn đề khác nhau cũng phải đối mặt do sự gia tăng mạnh mẽ của các tập dữ liệu dẫn đến dữ liệu lớn. Điều rất quan trọng là phải đưa ra các sáng kiến để bảo mật các tập dữ liệu và thông tin nhạy cảm này và cũng để đảm bảo rằng các công nghệ cơ sở dữ liệu được sử dụng một cách hiệu quả và có trách nhiệm [5].

Sự xuất hiện của mạng cảm biến không dây đã giúp tăng nhanh lượng dữ liệu được lưu trữ. Các mạng cảm biến không dây này liên tục truyền dữ liệu và

đã sinh ra thuật ngữ dữ liệu lớn, xu hướng được công nhận rộng rãi hiện nay. Thuật ngữ dữ liệu lớn này không chỉ liên quan đến khối lượng dữ liệu mà còn với tốc độ truyền tải cao và nhiều thông tin khác nhau khó thu thập, lưu trữ, đọc cập nhật và xóa bằng các giải pháp lưu trữ có sẵn khác nhau. Tất cả dữ liệu được thu thập và tạo ra bởi các cảm biến riêng lẻ không được coi là quan trọng và loại dữ liệu này được tạo và thu thập bởi các cảm biến khác nhau có khả năng tạo ra một lượng lớn khối lượng dữ liệu có thể khó xử lý cho các hoạt động CRUD cổ điển (Tạo, Đọc, Cập nhật và Xóa) một cách hiệu quả [13].

Khối lượng dữ liệu đang tràn ngập với tốc độ rất cao và tăng gấp đôi sau mỗi 18 tháng [9]. Các công cụ và công nghệ khác nhau có sẵn để nắm bắt và phân tích thông tin nhưng các tổ chức đang cố gắng đưa việc sử dụng dữ liệu lên cấp độ mới để đạt được hiệu suất và tính khả dụng tốt hơn theo cách tiết kiệm chi phí đang trở thành một thách thức lớn. Tốc độ tăng sẽ tăng nhanh hơn trong những tháng tiếp theo do việc sử dụng các thiết bị IoT liên tục gửi dữ liệu khiến lượng dữ liệu tràn ngập mỗi giây.

1.2 Internet of Things (IoT)

Chuỗi giá trị Internet of Things bao gồm tất cả các thiết bị thông minh và được kết nối. IoT là thuật ngữ đề cập đến xu hướng ngày càng tăng của việc sử dụng các cảm biến trong các thiết bị và vật thể để làm cho chúng có khả năng giao tiếp và gửi hoặc nhận thông tin [7]. Các lý do có thể khác nhau đối với việc các tổ chức ngày càng sử dụng nhiều cảm biến trong thiết bị. Cảm biến được sử dụng từ các thiết bị nhỏ, ví dụ: điện thoại thông minh cho đến loại lớn hơn như xe cộ, v.v. Cảm biến có thể khác nhau về loại, hình dạng và kích thước. Trong một động cơ ô tô, có nhiều loại cảm biến khác nhau, ví dụ như cảm biến nhiệt độ trong động cơ tạo ra cảnh báo nếu nhiệt độ của động cơ vượt qua ngưỡng cài đặt để thông báo cho người lái xe rằng có điều gì đó không ổn.

Có những ngành công nghiệp khác cũng được hưởng lợi từ cảm biến. Một số cảm biến có khả năng gửi dữ liệu đến nhà sản xuất để thông báo cho họ về lỗi hoặc các vấn đề trong thiết bị của họ để có thể giải quyết. Trong tương lai, việc sử dụng các cảm biến sẽ tăng lên rất nhiều. Cisco đã dự đoán và đưa ra một báo cáo vào năm 2011 rằng sẽ có 25 tỷ thiết bị trên Internet vào năm 2015, con số này sẽ tăng lên gấp đôi (50 tỷ) vào năm 2020 [16].

Ngày nay, Big Blue, biệt danh của IBM, đang đưa công nghệ nhỏ bé đó vào hoạt động, phát triển một cảm biến khí đa ứng dụng có thể giúp các sân bay phát hiện và theo dõi các mối đe dọa sinh hóa, xác định xem miếng bít tết trong tủ lạnh có bị hư hỏng hay không. chẩn đoán ung thư vú và các bệnh khác đơn giản bằng cách phân tích hơi thở của con người.

Cảnh quan công nghệ đang tiếp tục phát triển nhanh chóng. Facebook chạm mốc 500 triệu người dùng, người dùng điện thoại di động đạt 4 tỷ người, và lượng người dùng internet trên môi trường di động cũng đạt 450 triệu người. Tương tự như vậy, công nghệ thông tin cũng đã thay đổi cách triển khai như tăng cường sử dụng điện toán đám mây và ảo hóa. Sự thay đổi nhanh chóng của môi trường công nghệ cũng đã thúc đẩy một thuật ngữ Internet of Things (IoT) có khả năng nắm bắt, tính toán, giao tiếp và cộng tác. Các thiết bị IoT này được nhúng với các cảm biến, bộ truyền động và khả năng giao tiếp và sẽ sớm có thể gửi một lượng lớn dữ liệu khổng lồ trên quy mô lớn [9].

Với sự xuất hiện của tiêu chuẩn an toàn, một số công nghệ cốt lõi cho IoT đang được sử dụng rộng rãi hơn. Các công ty bảo hiểm ô tô ở Châu Âu và Hoa Kỳ đang thử nghiệm và lắp đặt các cảm biến trên xe của khách hàng để có thể cảm nhận được hành vi lái xe của tài xế để tính phí theo đó. Ngoài ra, một số nhà sản xuất ô tô hạng sang đang sử dụng các cảm biến tiên tiến để phản ứng tự động trong tình huống sắp xảy ra tai nạn. [9]

Các ứng dụng của IoT:

- IoT Cảm biến

Cảm biến IoT bao gồm các cảm biến thủ công hoặc kỹ thuật số được kết nối với bảng mạch như Arduino Uno hoặc Raspberry Pi 2. Bảng mạch có thể được lập trình để đo một loạt dữ liệu được thu thập từ thiết bị cảm biến như carbon monoxide, nhiệt độ, độ ẩm, áp suất, độ rung và chuyển động.

Điều khác biệt giữa các cảm biến IoT với các cảm biến đơn giản là chúng không chỉ có thể thu thập dữ liệu tại các môi trường vật lý khác nhau mà còn gửi dữ liệu đến các thiết bị được kết nối.

Hình 1.2 Cảm biến [27]

Các cảm biến IoT cho phép kiểm soát dữ liệu liền mạch thông qua tự động hóa cung cấp thông tin chi tiết hữu ích. Chúng có thể được các doanh nghiệp sử dụng để bảo trì dự đoán, nâng cao hiệu quả và giảm chi phí.

- Lưới điện thông minh:

  Lưới điện thông minh là một ứng dụng công nghiệp khác của IoT. Lưới điện cho phép theo dõi thời gian thực các dữ liệu liên quan đến cung và cầu điện. Nó liên quan đến việc áp dụng trí thông minh máy tính để quản lý hiệu quả các nguồn lực.

  Các công ty tiện ích có thể sử dụng các công nghệ lưới điện thông minh IoT để quản lý cúp điện hiệu quả hơn. Họ có thể sử dụng công nghệ để xác định phân bố tải và cải thiện độ tin cậy. Công nghệ này cũng có thể hỗ trợ phát hiện lỗi và sửa chữa.

Hình 1.3 Lưới điện thông minh [27]

Với lưới điện thông minh, các tiện ích có thể kết nối tất cả các tài sản của họ bao gồm cả công tơ và trạm biến áp. Việc áp dụng các công nghệ IoT vào hệ

sinh thái lưới điện cho phép các công ty tiện ích thực hiện quyền kiểm soát tốt hơn đối với cơ sở hạ tầng và tài nguyên điện. Hơn nữa, chúng cho phép người tiêu dùng tiếp cận năng lượng với chất lượng tốt hơn.

- Hệ thống chăm sóc sức khỏe:

IoT có rất nhiều ứng dụng trong ngành chăm sóc sức khỏe. Công nghệ này có thể được sử dụng để cung cấp các dịch vụ y tế chất lượng cao bằng các thiết bị y tế thông minh.

Còn được gọi là Internet of Medical Things (IoMT), công nghệ này có thể giúp theo dõi và hỗ trợ các dữ liệu quan trọng có thể giúp đưa ra các quyết định lâm sàng. Với các thiết bị y tế IoT, các dịch vụ y tế có thể được phổ biến rộng rãi hơn.

Các thiết bị y tế IoT có thể giúp theo dõi bệnh nhân từ xa theo thời gian thực. Các thiết bị này có thể thông báo trường hợp khẩn cấp như lên cơn suyễn, suy tim, v.v., ngay lập tức cho bác sĩ. Điều này có thể giúp cứu sống nhiều cá nhân.

Hình 1.4 Y tế thông minh [27]

Các thiết bị IoT có thể thu thập dữ liệu chăm sóc sức khỏe bao gồm huyết áp, lượng đường, oxy và cân nặng. Dữ liệu được lưu trữ trực tuyến và bác sĩ có thể truy cập bất cứ lúc nào. IoT tự động hóa quy trình làm việc bằng cách cho phép cung cấp các dịch vụ chăm sóc sức khỏe hiệu quả cho bệnh nhân.

- Đầu đọc mã vạch thông minh

Máy đọc mã vạch IoT có thể giúp quản lý hàng tồn kho tốt hơn cho các nhà bán lẻ. Các đầu đọc hỗ trợ xử lý tín hiệu kỹ thuật số dựa trên AI. Những

thiết bị này có thể tối ưu hóa hoạt động của nhiều lĩnh vực bao gồm bán lẻ, hậu cần, kho hàng, v.v.

Đầu đọc thẻ thanh dựa trên IoT có tính năng kết nối dữ liệu đám mây để kết nối với các hệ thống khác. Sử dụng đầu đọc mã vạch được kết nối sẽ dễ dàng hơn trong quá trình quản lý hàng tồn kho.

Hình 1.5 Đầu đọc mã vạch thông minh [27]

Đầu đọc mã vạch IoT có thể được kết hợp vào xe đẩy hàng. Người đọc sử dụng cảm biến dựa trên AI để phát hiện sản phẩm khi chúng được đánh rơi hoặc lấy ra khỏi giỏ hàng. Đầu đọc có thể chuyển dữ liệu sang máy tính tự động và điều đó có thể tiết kiệm rất nhiều thời gian trong quá trình thanh toán, dẫn đến cải thiện trải nghiệm cho khách hàng.

1.3 Các công đánh giá hiệu năng

Có một số công cụ đánh giá phổ biến được thiết kế chủ yếu cho các hệ hệ quản trị cơ sở dữ liệu truyền thống bao gồm bộ đánh giá TPC [10] và Wisconsin [7], cả hai công cụ sẽ được mô tả bên dưới. Tuy nhiên, Binning et al. [5] lập luận rằng các công cụ đánh giá truyền thống không phù hợp để phân tích các dịch vụ đám mây, và đề xuất một số ý tưởng giúp cải thiện khả năng mở rộng và đặc tính chịu lỗi của điện toán đám mây. Tiếp theo, một số công cụ đo lường được thiết kế đặc biệt dành cho các hệ quản trị cơ sở dữ liệu NoSQL.

1.3.1 Các công cụ đánh giá hiệu năng truyền thống

Mỗi tiêu chuẩn đo lường TPC được thiết kế để mô hình hóa một ứng dụng cụ thể trong thế giới thực bao gồm các ứng dụng xử lý giao dịch (OLTP) với tiêu chuẩn TPC-C và TPC-E, hệ thống hỗ trợ quyết định với tiêu chuẩn TPC-D và TPC-H, hệ quản trị cơ sở dữ liệu được lưu trữ trong môi trường ảo hóa với

tiêu chuẩn TPC-VMS (bao gồm tất cả bốn điểm chuẩn vừa được đề cập) và gần đây nhất là tiêu chuẩn dữ liệu lớn TPCx-HS cho các lớp Hadoop [10].

Mặt khác, tiêu chuẩn Wisconsin đánh giá các thành phần cơ bản của các hệ quản trị cơ sở dữ liệu quan hệ, như một cách để so sánh các hệ quản trị cơ sở dữ liệu khác nhau. Mặc dù không còn phổ biến như trước đây, nhưng nó vẫn là một đánh giá mạnh mẽ của người dùng về các hoạt động cơ bản mà hệ quản trị cơ sở dữ liệu quan hệ phải cung cấp, đồng thời nêu những bất thường về hiệu suất. Tiêu chuẩn hiện được sử dụng để đánh giá các đặc điểm như tăng kích thước, tăng tốc và mở rộng quy mô của các hệ quản trị cơ sở dữ liệu song song DBMS [7].

BigBench [11] là công cụ đo lường các đặc điểm kỹ thuật dựa trên các tiêu chuẩn đã tồn tại trên các công cụ đánh giá các loại dữ liệu lớn như YCSB, HiBench, Big data Benchmark, TPC-xHC, TPC-DS, GridMix, PigMix. Nó hỗ trợ dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc. Một phần cấu trúc của lược đồ BigBench được áp dụng từ mô hình dữ liệu TPC-DS được mô tả một sản phẩm riêng lẻ, và được mở rộng thêm với dữ liệu bán và không có cấu trúc. Khối lượng dữ liệu thô có thể được thay đổi động dựa trên hệ số tỷ lệ, nhưng tần suất cập nhật dữ liệu bị bỏ qua khi tạo dữ liệu điểm chuẩn.

1.3.2 Các công cụ đánh giá hiệu năng NoSQL

Năm 2009, Pavlo et al. [13] đã tạo ra một công cụ đo tiêu chuẩn được thiết kế để chuẩn hóa hai cách tiếp cận phân tích dữ liệu quy mô lớn: Map-Reduce và hệ quản trị cơ sở dữ liệu song song (DBMS) trên các cụm máy tính lớn. Map-Reduce là một mô hình lập trình và triển khai liên kết, song song thực hiện tính toán tập dữ liệu lớn trên các cụm quy mô lớn, xử lý các lỗi và lập lịch giao tiếp giữa các máy để sử dụng mạng và đĩa một cách hiệu quả [12]. Các hệ quản trị cơ sở dữ liệu song song đại diện cho các hệ quản trị cơ sở dữ liệu quan hệ có khả năng mở rộng theo chiều ngang để mang lại kích thước tập dữ liệu lớn.

Apache JMeter

Apache JMeter [21] là một công cụ dựa trên java mã nguồn mở được sử dụng để đánh giá nhiều công nghệ và dịch vụ khác nhau, công cụ bao gồm các hệ quản trị cơ sở dữ liệu, máy chủ web, máy chủ FTP, kiểm tra hiệu suất trang web và nhiều công cụ khác. JMeter đưa ra các lựa chọn thử nghiệm để xác định khả năng đáp ứng, độ tin cậy, khả năng mở rộng, thông lượng và khả năng

tương tác của một hệ thống hoặc bất kỳ ứng dụng nào với một khối lượng công việc được cung cấp.

JMeter có thể được sử dụng để chạy một hoặc hàng triệu hoạt động đồng thời. Ứng dụng JMeter bao gồm một máy chủ JMeter có thể xử lý và triển khai kế hoạch thử nghiệm. Khi một kế hoạch kiểm tra được gửi đến máy chủ JMeter, nó sẽ phản hồi và xử lý kế hoạch và gửi kết quả trở lại máy chủ lưu trữ từ nơi kế hoạch kiểm tra đã được gửi. Bằng cách này, JMeter-server có khả năng chạy kế hoạch thử nghiệm với một máy chủ từ nhiều máy khách cùng một lúc.

Sau khi kế hoạch thử nghiệm được thực hiện, kết quả của thử nghiệm có thể được thu thập tại tổng thể. JMeter có khả năng vẽ biểu đồ dựa trên kết quả nhưng để hình dung sâu hơn, tập dữ liệu được tạo ra từ kết quả của kế hoạch thử nghiệm có thể được lưu ở định dạng XML hoặc CSV. Các tệp XML hoặc CSV này của các bộ dữ liệu lớn hơn có thể được trực quan hóa và đồ thị có thể được vẽ dựa trên các bộ dữ liệu này bằng cách sử dụng các công cụ phân tích khác nhau (ví dụ: RStudio) để kiểm tra sâu [21].

YCSB

YCSB (Yahoo Cloud Serving Benchmark) là một framework mã nguồn mở để đánh giá và so sánh hiệu năng các hệ thống dữ liệu lớn được phát triển bởi Yahoo. YCSB có khả năng phân tích hiệu suất của các hệ hệ quản trị cơ sở dữ liệu thế hệ mới như NoSQL ngay cả trong môi trường có tài nguyên hạn chế. YCSB được sử dụng để tạo ra các kịch bản kiểm thử dựa trên một số hệ quản trị cơ sở dữ liệu và ghi kết quả dưới dạng báo cáo.

YCSB là một mô-đun và mỗi hệ quản trị cơ sở dữ liệu nó sử dụng có mô-đun ứng dụng khách được viết bằng java. Mô-đun này chứa các hoạt động CRUD cổ điển (Tạo, Đọc, Cập nhật, Xóa) thường được sử dụng trong hệ quản trị cơ sở dữ liệu. YCSB chứa một số khối lượng công việc được xác định trước để thực hiện đánh giá nhưng tùy thuộc vào các trường hợp sử dụng để đo lường hiệu suất của hệ quản trị cơ sở dữ liệu. YCSB cho phép người sử dụng tạo và tùy chỉnh kịch bản thử nghiệm để có được kết quả chính xác và thực tế hơn cho các trường hợp sử dụng được yêu cầu.

Khung YCSB được thiết kế để hỗ trợ so sánh một số hệ quản trị cơ sở dữ liệu NoSQL như Cassandra, HBase, Riak, v.v. Đây là một hệ thống điểm chuẩn nổi tiếng được thiết kế ban đầu để đánh giá trực tiếp các hệ quản trị cơ sở dữ liệu

không hỗ trợ ACID. YCSB bao gồm một trình tạo khối lượng công việc và một giao diện cơ sở dữ liệu cơ bản, có thể dễ dàng mở rộng để hỗ trợ các hệ quản trị cơ sở dữ liệu quan hệ hoặc NoSQL khác nhau. Nó cung cấp sáu khối lượng công việc được xác định trước, mô phỏng một ứng dụng OLTP trên đám mây (hoạt động đọc và cập nhật). Các chỉ số được báo cáo là thời gian thực thi và thông lượng (hoạt động trên giây). YCSB thực hiện một số giải pháp NoSQL bằng mô hình dữ liệu đơn giản nhưng sử dụng phần cứng hiệu suất cao.

Không có tiêu chí cụ thể để đánh giá các hệ quản trị cơ sở dữ liệu cảm biến cụ thể. Bộ dữ liệu thế giới thực, được tạo bởi Big Data Generator Suite, không bao gồm dữ liệu cảm biến có ý nghĩa về thời gian - vị trí nói riêng. Mặc dù có mối tương quan giữa mạng xã hội và dữ liệu cảm biến, nhưng vẫn có sự khác biệt giữa bản chất cơ bản của cả hai. Với các luồng dữ liệu khổng lồ là dữ liệu chuỗi thời gian không cố định, được thu thập ở các khoảng thời gian rời rạc, cần phải tìm ra giải pháp tốt nhất để lưu trữ và truy xuất dữ liệu cảm biến. Đặc biệt phải chú ý đến khả năng chèn một lượng cực lớn dữ liệu trong thời gian dài. Các thử nghiệm phải được thực hiện liên quan đến cả việc lưu trữ dữ liệu và tổng hợp dữ liệu trong các điều kiện cụ thể.

Công cụ YCSB có thể được sử dụng để đánh giá hệ quản trị cơ sở dữ liệu NoSQL bằng các phương thức sau:

- read () - đọc một bản ghi từ hệ quản trị cơ sở dữ liệu, và trả về một tập trường cụ thể hoặc tất cả các trường.
- insert () - chèn một bản ghi vào hệ quản trị cơ sở dữ liệu.
- update () - cập nhật một bản ghi duy nhất trong hệ quản trị cơ sở dữ liệu, thêm hoặc thay thế các trường cụ thể.
- delete () - xóa một bản ghi trong hệ quản trị cơ sở dữ liệu.
- scan () - thực hiện quét phạm vi, đọc một số lượng bản ghi cụ thể bắt đầu từ một khóa bản ghi nhất định.

Các thao tác này khá đơn giản, đại diện cho các thao tác "CRUD" tiêu chuẩn như: Tạo, Đọc, Cập nhật, Xóa, với các thao tác đọc để đọc một bản ghi hoặc để quét bản ghi. Các API của YCSB ánh xạ tốt với các API của nhiều hệ thống khác nhau.

YCSB là công cụ hỗ trợ kiểm tra hiệu năng khá mạnh, đặc biệt hỗ trợ nhiều loại cơ sở dữ liệu lớn khác nhau. Phù hợp với các nghiệp vụ kiểm thử hiệu

năng của các khối lượng công việc đơn giản (có thể triển khai trên nhiều hệ quản trị khác nhau) và đưa ra đề xuất lựa chọn hệ quản trị phù hợp nhất với ứng dụng dữ liệu lớn cần kiểm thử.

Như vậy, trong chương này tôi đã đưa ra một số thông tin cơ bản về dữ liệu lớn, những lợi ích mà dữ liệu lớn đem lại cho chúng ta. Bên cạnh đó cũng có những thách thức khi triển khai và lưu trữ dữ liệu lớn. Luận văn cũng giới thiệu một số công cụng đo hiệu năng các hệ quản trị có sở dữ liệu truyền thống và các hệ quản trị cơ sở dữ liệu NoSQL.

Chương 2: Một số hệ quản trị cơ sở dữ liệu NoSQL

Nội dung của chương nay sẽ trình bày những thông tin cơ bản về các đăc tính kỹ thuật của một số hệ quản trị cơ sở dữ liệu NoSQL và phân loại các hệ quản trị phổ biến nhất hiện nay.

Do sự phát triển của Internet và điện toán đám mây, thuật ngữ Internet of Things (IoT) ngày nay được sử dụng rộng rãi và nó đã làm tăng đáng kể khối lượng dữ liệu, do đó cần phải có các giải pháp để các hệ quản trị cơ sở dữ liệu có khả năng lưu trữ và xử lý dữ liệu lớn hiệu quả hơn và hiệu quả [12]. Các giải pháp của một số hệ quản trị cơ sở dữ liệu được yêu cầu để cung cấp hiệu suất cao cho các hoạt động CRUD cổ điển mà cơ sở dữ liệu truyền thống đang gặp phải thách thức khi lưu trữ và truy vấn dữ liệu người dùng động. Trong trường hợp này, hệ quản trị cơ sở dữ liệu NoSQL tuyên bố là hiệu quả hơn và nhanh hơn so với RDBMS trong các trường hợp sử dụng khác nhau và đặc biệt khi thảo luận về các kho dữ liệu lớn nơi dữ liệu tăng nhanh và liên tục. Do sự xuất hiện của các ứng dụng và công nghệ mới, hệ quản trị cơ sở dữ liệu được yêu cầu để đáp ứng các nhu cầu sau [12].

- Đọc và ghi đồng thời cao với độ trễ thấp.
- Yêu cầu truy cập và lưu trữ dữ liệu lớn hiệu quả.
- Khả năng mở rộng cao và tính sẵn sàng cao.
- Giảm chi phí quản lý và vận hành.

Mặc dù hệ quản trị cơ sở dữ liệu quan hệ đã cung cấp giải pháp lưu trữ dữ liệu và các nhà cung cấp đã cố gắng cập nhật hệ thống cơ sở dữ liệu để đáp ứng những nhu cầu mới nhất nhưng các hệ thống cơ sở dữ liệu truyền thống vẫn không cung cấp giải pháp lưu trữ hiệu quả và hiệu quả theo yêu cầu của người sử dụng do các nguyên nhân sau [12]:

- Đọc và viết chậm
- Năng lực hạn chế
- Khó mở rộng

NoSQL là viết tắt của Not Only SQL và thuật ngữ này được sử dụng cho các cơ sở dữ liệu thay thế cho RDBMS (ví dụ: Oracle, MS SQL Server và IBM, v.v.). Hệ quản trị cơ sở dữ liệu NoSQL có khả năng lưu trữ lượng dữ liệu lớn hơn. Hệ cơ sở dữ liệu NoSQL truy xuất thông tin nhanh chóng và di động. Do đó, một tính năng quan trọng khác của hệ quản trị cơ sở dữ liệu NoSQL là mã

nguồn mở, mã của nó có sẵn cho mọi người và có thể được sửa đổi và tuân thủ theo yêu cầu. Hệ cơ sở dữ liệu NoSQL hiển thị hiệu suất cao theo cách tuyến tính và có thể mở rộng theo chiều ngang. NoSQL không tổ chức dữ liệu của nó trong các bảng có liên quan.

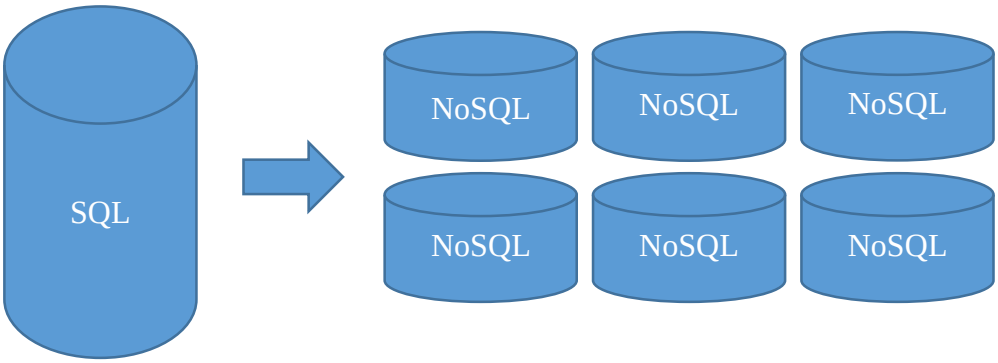
Mặc dù hệ quản trị cơ sở dữ liệu quan hệ truyền thống được sử dụng rộng rãi trong nhiều thập kỷ và các nhà cung cấp cơ sở dữ liệu không ngừng cố gắng cải tiến chúng để xử lý và hỗ trợ khối lượng lớn dữ liệu. Tuy nhiên, loại công nghệ của hệ quản trị cơ sở dữ liệu mới được gọi là NoSQL có khả năng hỗ trợ, xử lý khối lượng dữ liệu lớn hơn với khả năng truy cập nhanh hơn và chi phí lưu trữ ít hơn. Hệ quản trị cơ sở dữ liệu NoSQL được coi là một thành phần quan trọng của Dữ liệu lớn khi nói đến việc truy xuất và lưu trữ một lượng lớn dữ liệu. Một ví dụ khác về trường hợp sử dụng của NoSQL là lưu trữ một lượng lớn dữ liệu trong Facebook (chúng tiếp tục tăng lên mỗi giây). NoSQL ngày càng trở nên phổ biến do dung lượng lưu trữ cao và được sử dụng rộng rãi hiện nay, vd:

- Google (BigTable, LevelDB)
- Linkedin (Voldemort)
- Facebook (Cassandra)
- Twitter (Hadoop / Hbase, FlockDB, Cassandra)
- Netflix (SimpleDB, Hadoop, HBase, Cassandra)
- CERN (CouchDB)

Hệ quản trị cơ sở dữ liệu NoSQL sử dụng định lý BASE để nhất quán dữ liệu trong khi RDBMS sử dụng định lý ACID. Một ưu điểm khác của hệ quản trị cơ sở dữ liệu NoSQL so với RDBMS là NoSQL có thể mở rộng theo cả chiều ngang và chiều dọc trong khi RDBMS chỉ có thể mở rộng theo chiều dọc. [10]

Hệ quản trị cơ sở dữ liệu quan hệ hiển thị cơ chế chạy và hoạt động trên một máy duy nhất, do đó cần một máy mạnh và lớn để mở rộng quy mô. Trong trường hợp này vì toàn bộ hệ quản trị cơ sở dữ liệu phụ thuộc vào một máy duy nhất và nếu máy đó gặp sự cố, thì toàn bộ hệ quản trị cơ sở dữ liệu sẽ đi xuống. Giải pháp cho vấn đề này là mua một số máy nhỏ thay vì một máy đơn lẻ và tạo một cụm gồm nhiều máy để lưu trữ dữ liệu. Quá trình này được coi là rẻ hơn và có thể mở rộng theo chiều ngang. Trong trường hợp này nếu một máy gặp sự cố thì các máy khác duy trì độ tin cậy của cụm khá cao. Đây là cơ chế được hiển thị

bởi cơ sở dữ liệu NoSQL và đó là một lý do mà hệ quản trị cơ sở dữ liệu NoSQL ngày nay trở nên khá phổ biến [16].



Hình 2.1 Suy giảm sự thống trị của SQL.

Một số tính năng quan trọng theo sau của cơ sở dữ liệu NoSQL như sau:

A. ACID free

Hiệu suất và khả năng mở rộng tốt hơn trong NoSQL đạt được bằng cách hy sinh khả năng tương thích ACID. ACID là viết tắt của tính nguyên tử, nhất quán, độc lập và bền vững. Về cơ bản, khái niệm ACID xuất phát từ môi trường SQL nhưng do yếu tố nhất quán, các giải pháp NoSQL tránh sử dụng khái niệm ACID. Hệ quản trị cơ sở dữ liệu NoSQL dựa trên các hệ thống phân tán và dữ liệu được lan truyền đến các máy khác nhau trong cụm và nó được yêu cầu để duy trì tính nhất quán. Ví dụ, nếu có sự thay đổi trong một bảng, thì bắt buộc phải thực hiện thay đổi trong tất cả các máy có dữ liệu. Có thể đạt được sự nhất quán nếu thông tin về quá trình cập nhật lan truyền ngay lập tức qua toàn bộ hệ thống, nếu không, sự không nhất quán được thực hiện và theo cách này, khái niệm ACID tạo ra rắc rối cho các giải pháp NoSQL.

B. BASE

BASE là đảo ngược của ACID và thuật ngữ này là viết tắt của trạng thái Cơ bản, Sẵn sàng, Mềm mỏng và tính nhất quán cuối cùng. Sử dụng sao chép và phân bổ để giảm khả năng dữ liệu không có sẵn và sử dụng sharding. Do đó, hệ thống luôn sẵn sàng ngay cả khi các mạng con của dữ liệu bị hỏng và không khả dụng trong một khoảng thời gian ngắn. Do đó, nói chung tính khả dụng của BASE đạt được thông qua việc hỗ trợ sự cố từng phần mà không có sự cố toàn bộ hệ thống. Ví dụ: nếu chúng ta thảo luận về hệ quản trị cơ sở dữ liệu ngân hàng trong các ngân hàng và hai người cố gắng truy cập vào cùng một tài khoản

ngân hàng từ hai địa điểm khác nhau thì không chỉ cần cập nhật dữ liệu kịp thời mà còn yêu cầu một số hệ quản trị cơ sở dữ liệu thời gian thực. Một số ví dụ khác về tình huống tương tự có thể là đặt vé trực tuyến và các nền tảng mua sắm trực tuyến.

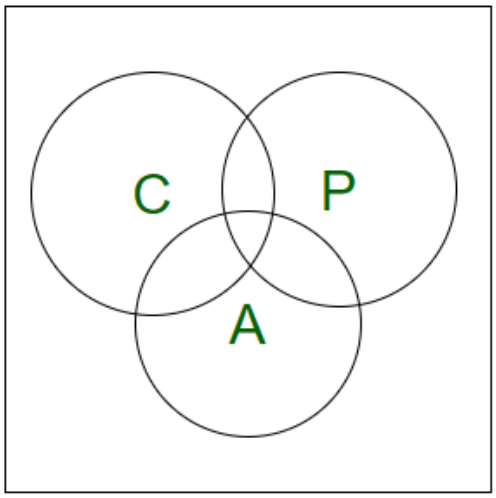
ACID	BASE
Tính nhất quán cao cho mức ưu tiên cao nhất của giao dịch	Mức độ sẵn có và mức ưu tiên cao nhất mở rộng
Tính khả dụng ít quan trọng hơn	Tính nhất quán yếu
Bi quan (pessimistic)	Lạc quan (optimistic)
phân tích chặt chẽ	Hiệu suất tốt
cơ chế phức tạp	đơn giản và nhanh chóng

Hình 2.2 So sánh ACID và BASE [29]

C. CAP

CAP là viết tắt của Tính nhất quán, Tính khả dụng và Dung sai phân vùng. Định lý CAP dựa trên ba nguyên tắc này.

- C viết tắt "Consistency" Tính nhất quán có nghĩa dữ liệu phải nhất quán và có sẵn trên tất cả các máy phải giống nhau về mọi mặt và quá trình cập nhật phải chạy trên tất cả các máy thường xuyên.
- A viết tắt "Availability" Tính khả dụng nghĩa là dữ liệu phải luôn sẵn sàng cho khách hàng và phải có thể truy cập bất cứ lúc nào.
- P viết tắt "Partition tolerance" Dung sai phân vùng nếu do lỗi hệ thống và lỗi trong các nút, các hệ quản trị cơ sở dữ liệu phải hoạt động tốt bất chấp các phân vùng mạng vật lý, tức là dung sai phân vùng.



Hình 2.3: Nguyên lý định lý CAP

Có nhiều loại cơ sở dữ liệu NoSQL khác nhau và mỗi loại có một bộ tính năng và đặc điểm riêng, và những điều này dẫn đến sự khác biệt về hiệu suất.

Các loại cơ sở dữ liệu NoSQL khác nhau:

- Key-values Stores
- Column Family Stores
- Document store Databases
- Graph Databases

Các hệ quản trị cơ sở dữ liệu NoSQL cung cấp hiệu suất tăng nhưng một số nhà nghiên cứu nghi ngờ về tính nhất quán của dữ liệu.

Mô hình dữ liệu	Hiệu suất	Khả năng mở rộng	Linh họa	Phức tạp	Chức năng	Hệ quản trị NoSQL
Khóa/giá trị	Cao	Cao	Cao	Vừa phải	Mảng liên kết	Redis, Voldemort
Cột	Cao	Cao	Vừa phải	Thấp	Cơ sở dữ liệu cột	Hbase, Cassandra
Tài liệu	Cao	Bất định (Cao)	Cao	Thấp	Hướng đối tượng	MongoDB, SimpleDB
Đồ thị	Bất định	Bất định	Cao	Cao	Lý thuyết đồ thị	OrientDB, Neo4J,

Bảng 2.1: Phân loại hệ quản trị cơ sở dữ liệu NoSQL

2.1. Phân loại cơ sở dữ liệu NoSQL

2.1.1. Loại cơ sở dữ liệu khóa – giá trị

Hệ quản trị cơ sở dữ liệu NoSQL đơn giản nhất chính là Key/Value stores. Nó đơn giản nhất là vì những API của nó đơn giản, những triển khai thực tế của NoSQL thường rất phức tạp. Hầu hết Khóa/Giá trị thường có những API sau:

```
void Put(string key, byte[] data);
byte[] Get(string key);
void Remove(string key);
```

Khóa	Giá trị
ten	Anh
ho	Nguyễn
diachi	Bắc Ninh

Hình 2.4: Loại cơ sở dữ liệu khóa – giá trị

Với khóa-giá trị thì việc truy xuất, xóa, cập nhật giá trị thực đều thông qua key tương ứng. Giá trị được lưu dưới dạng BLOB (Binary large object). Xây dựng một khóa/giá trị rất đơn giản và mở rộng chúng cũng rất dễ dàng. Khóa/giá trị có hiệu suất rất tốt bởi vì mô hình truy cập dữ liệu trong khóa/giá trị được tối ưu hóa tối đa. Khóa/giá trị là cơ sở cho tất cả những loại cơ sở dữ liệu NoSQL khác.

Khóa/giá trị rất hữu ích khi cần truy cập dữ liệu theo khóa. Ví dụ như k cần khi lưu trữ thông tin phiên giao dịch hoặc thông tin giỏ hàng của người dùng thì khóa/giá trị là một sự lựa chọn hợp lý bởi vì nhờ vào id của người dùng nó có thể nhanh chóng lấy được các thông tin liên quan trong phiên giao dịch hoặc giỏ hàng của người dùng đó. Giỏ mua hàng của Amazon chạy trên khóa/giá trị (Amazon Dynamo). Vì thế có thể thấy rằng khóa/giá trị store có khả năng mở rộng cao. Amazon Dynamo Paper là một ví dụ tốt nhất về kiểu dữ liệu khóa/giá. Rhino DHT có khả năng mở rộng, chuyển đổi dự phòng, không cấu hình, là dạng khóa/giá trên nền tảng .Net.

2.1.2. Loại cơ sở dữ liệu cột

Loại cơ sở dữ liệu cột là hệ quản trị cơ sở dữ liệu phân tán cho phép truy xuất ngẫu nhiên/tức thời với khả năng lưu trữ một lượng cực lớn dữ liệu có cấu trúc. Dữ liệu có thể tồn tại dạng bảng với hàng tỷ bảng ghi và mỗi bảng ghi có thể chứa hàng triệu cột. Một triển khai từ vài trăm cho tới hàng nghìn nút dẫn đến khả năng lưu trữ hàng Petabytes dữ liệu nhưng vẫn đảm bảo hiệu suất cao.

Cơ sở dữ liệu cột được biết đến nhiều nhất thông qua sự triển khai BigTable của Google. Nhìn bên ngoài vào nó giống với hệ quản trị cơ sở dữ liệu quan hệ nhưng thực sự thì có sự khác biệt rất lớn từ bên trong. Một trong những khác biệt đó chính là việc lưu trữ dữ liệu theo cột so với việc lưu trữ dữ liệu theo dòng (trong hệ quản trị cơ sở dữ liệu quan hệ). Sự khác biệt lớn nhất chính là bản chất của nó. Chúng ta không thể áp dụng cùng một giải pháp mà chúng ta sử dụng trong hệ quản trị cơ sở dữ liệu quan hệ vào trong loại cơ sở dữ liệu họ cột. Đó là bởi vì dữ liệu họ cột là phi quan hệ. Các khái niệm sau đây rất quan trọng để hiểu được hệ quản trị cơ sở dữ liệu column family làm việc như thế nào:

- Column family (cột quan hệ)
- Super column (siêu cột)
- Column (cột)

Cột quan hệ: Một cột quan hệ là cách thức dữ liệu được lưu trữ trên đĩa cứng. Tất cả dữ liệu trong một cột sẽ được lưu trên cùng một file. Một họ cột có thể chứa nhiều cột hoặc một cột.

<table>
<tr><td rowspan="2">Dòng Id</td><td>Id 1</td><td>Id 2</td><td>Id 3</td><td rowspan="2">...</td></tr>
<tr><td>Cột giá trị 1</td><td>Cột giá trị 2</td><td>Cột giá trị 3</td></tr>
<tr><td colspan="5">⋮</td></tr>
</table>

Hình 2.5: Loại cơ sở dữ liệu cột

Siêu cột: Một siêu cột có thể được dùng như một kiểu từ điển. Nó là một cột có thể chứa những cột khác (mà không phải là siêu cột).

<table>
<tr><td rowspan="3">Dòng Id 1</td><td colspan="3">Siêu Id 1</td><td colspan="3">Siêu Id 2</td><td rowspan="3">…</td></tr>
<tr><td>Sub id 1</td><td>Sub id 2</td><td rowspan="2">…</td><td>Sub id 3</td><td>Sub id 4</td><td rowspan="2">…</td></tr>
<tr><td>Cột giá trị 1</td><td>Cột giá trị 1</td><td>Cột giá trị 3</td><td>Cột giá trị 4</td></tr>
<tr><td colspan="8">⋮</td></tr>
</table>

Hình 2.6: Loại cơ sở dữ liệu siêu cột

Cột: Một cột là một bộ gồm tên, giá trị và dấu thời gian (thông thường chỉ quan tâm tới khóa-giá trị).

Một số loại khóa/giá trị phổ biến:

- Key/value cache in RAM: memcached, Citrusleaf database, Velocity, Redis, Tuple space...
- Key/value save on disk: Memcachedb, Berkeley DB, Tokyo Cabinet, Redis...
- Eventually Consistent Key Value Store: Amazon Dynamo, Voldemort, Dynomite, KAI, Cassandra, Hibari, Project Voldemort…
- Ordered key-value store: NMDB, Memcachedb, Berkeley DB...
- Distributed systems: Apache River, MEMBASE, Azure Table Storage, Amazon Dynamo ...

2.1.3. Loại cơ sở dữ liệu tài liệu

Khái niệm trung tâm của cơ sở dữ liệu tài liệu là khái niệm "tài liệu". Về cơ bản thì hệ quản trị cơ sở dữ liệu tài liệu là một khóa-giá trị với giá trị nằm trong một định dạng được biết đến (không định dạng). Mỗi loại cơ sở dữ liệu tài liệu được triển khai khác nhau ở phần cài đặt chi tiết nhưng tất cả tài đều được đóng gói và mã hóa dữ liệu trong một số định dạng tiêu chuẩn hoặc mã hóa. Một số kiểu mã hóa được sử dụng bao gồm XML, YAML, JSON, và BSON, cũng như kiểu nhị phân như PDF và các tài liệu Microsoft Office (MS Word, Excel …).

Trên thực tế, tất cả cơ sở dữ liệu tài liệu đều sử dụng JSON(hoặc BSON) hoặc XML.

Các tài liệu bên trong một hệ quản trị cơ sở dữ liệu tài liệu thì tương tự nhau, nó gần giống với khái niệm "bản ghi" hay "dòng" trong hệ quản trị cơ sở dữ liệu quan hệ truyền thống nhưng nó ít cứng nhắc hơn. Tài liệu không bắt buộc phải tuân theo một lược đồ tiêu chuẩn cũng không cần phải có tất cả các thuộc tính, khóa tương tự nhau. Xem ví dụ dưới đây:

Tài liệu 1	Tài liệu 2

{
ten:"Bob",
diachi:"5 Oak St.",
sothich:"sailing"
}
	{
	ten:"Jonathan",
	diachi:"15 Hoan kiem",
	Children:[
	{ten:"AA",Age:10},
	{ten:"BB", Age:8},
	{ten:"CC", Age:5},
	{ten:"DD", Age:2}
	]
	}

Cả hai tài liệu trên có một số thông tin tương tự và một số thông tin khác nhau. Không giống như một cơ sở dữ liệu quan hệ truyền thống, nơi mỗi bản ghi (dòng) có cùng một tập hợp trường dữ liệu và các trường dữ liệu này nếu không sử dụng thì có thể được lưu trữ rỗng, còn trong hệ quản trị cơ sở dữ liệu tài liệu thì không có trường dữ liệu rỗng trong tài liệu. Hệ thống này cho phép thông tin mới được thêm vào mà không cần phải khai báo rõ ràng.

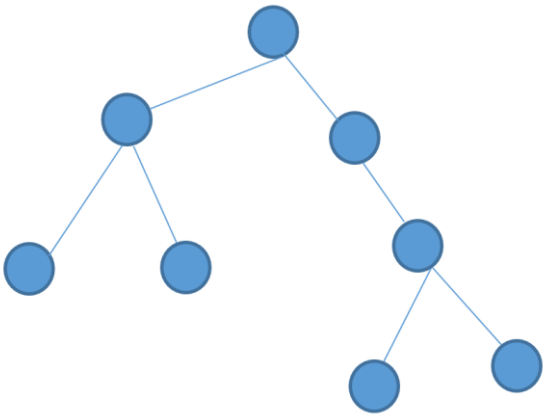
Các tài liệu được đánh dấu trong hệ quản trị cơ sở dũ liệu tài liệu thông qua một khóa duy nhất đại diện cho tài đó. Thông thường, khóa này là một chuỗi đơn giản. Trong một số trường hợp, chuỗi này có thể là một URI hoặc đường dẫn. Chúng ta có thể sử dụng khóa này để lấy tài liệu từ cơ sở dữ liệu. Thông

thường, cơ sở dữ liệu vẫn lưu lại một chỉ số trong khóa của tài liệu để tài liệu có thể được tìm kiếm nhanh chóng. Ngoài ra, cơ sở dữ liệu sẽ cung cấp một API hoặc ngôn ngữ truy vấn cho phép lấy các tài liệu dựa trên nội dung. Ví dụ, chúng ta muốn truy vấn lấy những tài liệu mà những tài liệu đó có tập trường dữ liệu nhất định với những giá trị nhất định.

Các hệ quản trị cơ sở dữ liệu tài liệu phổ biến là: BaseX, ArangoDB, Clusterpoint, Couchbase Server, CouchDB, eXist, FleetDB, Jackrabbit, Lotus Notes, MarkLogic, MongoDB, MUMPSDatabase, OrientDB, Apache Cassandra, Redis, Rocket U2, RavenDB…. Lưu ý: hầu hết cơ sở dữ liệu XML đều là triển khai của hệ quản trị cơ sở dữ liệu tài liệu. Một số XML trong danh sách các phổ biến là: BaseX, eXist, MarkLogic, Sedna.

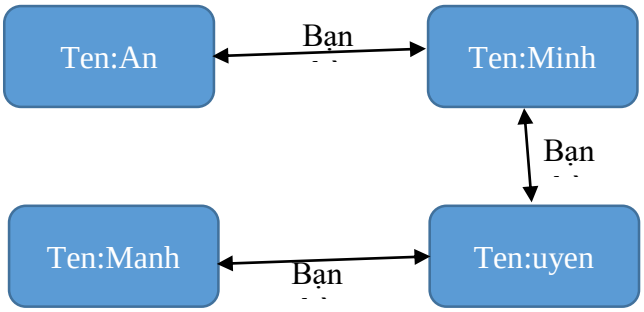
2.1.4. Loại cơ sở dữ liệu đồ thị

Hệ cơ sở dữ liệu đồ thị là một dạng cơ sở dữ liệu được thiết kế riêng cho việc lưu trữ thông tin đồ thị như cạnh, nút, các thuộc tính.



Hình 2.7: Loại cơ sở dữ liệu đồ thị

Chúng ta có thể nghĩ hệ quản trị cơ sở dữ liệu đồ thị như một hệ quản trị cơ sở dữ liệu tài liệu với các kiểu tài liệu đặc biệt và các mối quan hệ. Một ví dụ điển hình đó chính là mạng xã hội, có thể xem hình bên dưới:



Hình 2.8: Ví dụ về các nút trong một hệ quản trị cơ sở dữ liệu đồ thị

Trong ví dụ trên ta có 4 document và 3 mối quan hệ. Mối quan hệ trong cơ sở dữ liệu đồ thị thì có ý nghĩa nhiều hơn con trỏ đơn thuần. Một mối quan hệ có thể một chiều hoặc hai chiều nhưng quan trọng hơn là mối quan hệ được phân loại. Một người có thể liên kết với người khác theo nhiều cách, có thể là khách hàng, có thể là người trong gia đình…Mối quan hệ tự bản thân nó có thể mang thông tin. Trong ví dụ trên ta chỉ đơn giản lưu lại lại loại quan hệ và mức độ gần gũi (bạn bè, người trong gia đình, người yêu…).

Với hệ quản trị cơ sở dữ liệu đồ thị, chúng ta có thể thực hiện các hoạt động đồ thị. Một thao tác cơ bản nhất là điểm giao nhau. Ví dụ như nếu ta muốn biết những người trong thị trấn để cùng đi ăn uống thì đơn giản. Nhưng còn bạn bè gián tiếp thì sao, làm sao ta biết được họ. Sử dụng cơ sở dữ liệu đồ thị chúng ta có thể định nghĩa truy vấn sau:

```
new GraphDatabaseQuery
{
        SourceNode = ayende,
        MaxDepth = 3,
        RelationsToFollow = new[]{"đã biết", "Friend","Ex"},
        Where = node => node.Location == ayende.Location,
        SearchOrder = SearchOrder.BreadthFirst
}.Execute();
```

Chúng ta có thể thực hiện những truy vấn phức tạp hơn như lọc trên các thuộc tính quan hệ, xem xét trọng lượng của người đó... Cơ sở dữ liệu đồ thị thường được sử dụng để giải quyết các vấn đề về mạng. Trong thực tế, hầu hết các trang web mạng xã hội đều sử dụng một số hình thức của cơ sở dữ liệu đồ thị để làm những việc mà chúng ta đã biết như: kết bạn, bạn của bạn…

Một vấn đề đối với việc mở rộng cơ sở dữ liệu đồ thị là rất khó để tìm thấy một đồ thị con độc lập, có nghĩa là rất khó để ta phân tán graph database thành nhiều mảnh. Có rất nhiều nỗ lực nghiên cứu cho việc này nhưng chưa có bất kỳ giải pháp nào đáng tin cậy được đưa ra.

Một số sản phẩm tiêu biểu của cơ sở dữ liệu đồ thị là: Neo4J, Sones, AllegroGraph, Core Data, DEX, FlockDB, InfoGrid, OpenLink Virtuoso, ...

2.2. Một số hệ quản trị cơ sở dữ liệu NoSQL phổ biến

2.2.1. Hệ quản trị cơ sở dữ liệu Cassandra

Apache Cassandra là một hệ quản trị cơ sở dữ liệu NoSQL phân tán, ban đầu được phát triển bởi Facebook và trở thành công cụ nguồn mở năm 2008. Sau đó được chuyển giao cho Apache từ năm 2009. Cassandra lưu trữ dữ liệu bằng cách phân tán dữ liệu ra các nút khác nhau trong một cụn để đảm bảo việc xử lý dữ liệu nhanh chóng và an toàn dù có một hoặc một số nút xảy ra lỗi.

Apache Cassandra là một hệ quản trị cơ sở dữ liệu NoSQ, lưu trữ dữ liệu dưới dạng cột rộng bằng cách kết hợp cả dạng khóa-giá trị và dạng bảng. Thành phần chính của Cassandra là Keyspace với 3 thuộc tính sau:

– Yếu tố nhân bản: quy định số lượng nút trong cụm sẽ nhận bản copy của cùng một dữ liệu.

– Chiến lược nhân bản: quy định cách lưu trữ các bản sao, ví dụ như simple strategy, old network topology strategy, network topology strategy...

– Họ cột: dùng để mô tả cấu trúc của dữ liệu. Mỗi một họ cột có nhiều dòng và mỗi dòng lại có nhiều cột theo thứ tự nhất định (khác với cơ sở dữ liệu quan hệ, người dùng có thể tự do thêm cột vào bất kỳ lúc nào và các dòng không nhất thiết phải có cùng các cột). Thông thường mỗi Keyspace thường có ít nhất một hoặc nhiều họ cột.

Các đặc điểm nổi bật:

- Chấp nhận sai sót: dữ liệu được sao chép thành nhiều bản trên các máy chủ. Nếu chẳng may 1 máy chủ nào đó bị hỏng, thì vẫn có thể truy xuất dữ liệu trên các máy chủ khác.
- Tính co giãn: khả năng đọc/ghi tăng tuyến tính theo số lượng máy được thêm vào cụm máy mà không có thời gian chết hay sự gián đoạn ứng dụng đang chạy.
- Hướng cột: các RDBMS hướng dòng phải định nghĩa trước các cột trong các bảng. Đối với Cassandra không phải làm điều đó, đơn giản là thêm vào bao nhiêu cột cũng được tùy theo nhu cầu của bài toán.
- Tính sẵn sàng cao khi thực hiện tác vụ đọc/ghi, Cassandra có thể thực hiện trên bản sao gần nhất hoặc trên tất cả các bản sao. Điều này phụ thuộc vào thông số về mức độ nhất quán do thiết lập từ ban đầu.

- Tính điều hướng nhất quán: Đọc và ghi đưa ra một yêu cầu về tính nhất quán với việc "việc ghi không bao giờ bị lỗi".
- Hỗ trợ Map/Reduce: Cassandra có tích hợp thêm cả Hadoop đồng nghĩa với việc hỗ trợ map/reduce.

Có truy vấn theo ngôn ngữ riêng: CQL (viết tắt của Cassandra Query Language) là một thay thể của SQL – giống với các giao thức RPC truyền thống. Nó được điều khiển bởi Java và Python

2.2.2. Hệ quản trị cơ sở dữ liệu MongoDB

MongoDB là một mã nguồn mở và là một tập tài liệu dùng cơ chế NoSQL để truy vấn, nó được viết bởi ngôn ngữ C++. Chính vì được viết bởi C++ nên nó có khả năng tính toán với tốc độ cao chứ không giống như các hệ quản trị CSDL hiện nay. Mỗi một bảng dữ liệu trong SQL sử dụng thì trong MongoDB gọi là tập hợp. Mỗi một bản ghi trong MongoDB được gọi là tài liệu. Một bản ghi của MongoDB được lưu trữ dưới dạng tài liệu, nó được ghi xuống với cấu trúc trường và giá trị.

Điều đó có thể dễ dàng ép kiểu sang kiểu dữ liệu mảng để lập trình các ứng dụng một cách dễ dàng hơn. Nói một cách dễ hiểu thì mỗi một bản ghi của MongoDB là một mảng dữ liệu riêng biệt bao gồm các cặp key, value khác nhau do đó cách lưu trữ của MongoDB là phi cấu trúc dữ liệu.

MongoDB được sử dụng tốt nhất với nhu cầu cần truy vấn động, nếu muốn định nghĩa chỉ mục mà không cần các hàm map/reduce. Đặc biệt nếu cần tốc độ nhanh cho một hệ quản trị cơ sở dữ liệu lớn vì MongoDB ngoài tốc độ đọc nhanh ra thì tốc độ ghi của nó rất nhanh.

Các đặc điểm chính của mongoDB là:

- Các truy vấn Ad hoc: Mongo hỗ trợ việc tìm theo trường, khoảng kết quả tìm và tìm theo cú pháp. Các truy vấn có thể trả về các trường được qui định trong văn bản và cũng có thể bao gồm các hàm Javascript mà người dùng chưa định nghĩa.
- Đánh chỉ mục: Bất cứ một trường nào trong MongoDB đều được đánh chỉ mục (giống như chỉ mục bên RMDBs).
- Mô phỏng (nhân bản): Mongo hỗ trợ mô phỏng Master-slave. Một master có thể điều khiển việc đọc và ghi. Một slave tạo bản sao sữ liệu

từ master và chỉ được sử dụng cho việc đọc và sao lưu (không có quyền ghi). Slave có khả năng chọn ra một master mới nếu master cũ bị hỏng.

- Cân bằng tải: Mongo mở rộng theo chiều ngang bằng cách sử dụng Sharding. Các lập trình viên chọn các khóa chia sẻ nhằm xác định dữ liệu sẽ được phân tán như thế nào. Dữ liệu sẽ được tách thành các khoảng dựa vào khóa và phân tán dọc theo các Shard.
- Lưu trữ file: Mongo lưu trữ bằng file hệ thống, rất tốt cho việc cân bằng tải và nhân bản dữ liệu. Trong các hệ thống nhiều máy, các file được phân phối và được sao ra rất nhiều lần giữa các máy một cách trong suốt. Do đó rất hiệu quả trong việc tạo ra một hệ thống cân bằng tải và dung lỗi tốt.

2.2.3. Hệ quản trị cơ sở dữ liệu Redis

Redis là hệ quả trị cơ sở dữ liệu NoSQL, lưu trữ dữ liệu với dạng Khóa-Giá trị với nhiều tính năng được sử dụng rộng rãi. Nó có thể hỗ trợ nhiều kiểu dữ liệu như: strings, hashes, lists, sets, sorted. Đồng thời có thể cho phép viết kịch bản bằng ngôn ngữ Lua.

Redis ngoài tính năng lưu trữ Khóa-Giá trị trên RAM thì Redis còn hỗ trợ tính năng lưu trữ dữ liệu trên đĩa cứng cho phép có thể phục hồi dữ liệu khi hệ thống gặp sự cố. Redis hỗ trợ tính năng sao chép cho phép sao chép, đồng bộ giữa 2 CSDL Redis với nhau. Ngoài ra còn có tính năng cụm cũng đang được phát triển cho phép cân bằng tải.

Redis có những ưu điểm:

- Redis hỗ trợ thêm mới, cập nhật, xoá dữ liệu một cách nhanh chóng.

- Lưu trữ dữ liệu dạng KHÓA-GIÁ TRỊ.

- Dữ liệu được lưu trữ trên RAM giúp việc truy xuất dữ liệu một cách nhanh chóng. Ngoài ra, có thể cấu hình để Redis có thể lưu trữ dữ liệu trên ổ cứng.

- Tụ cấu hình cho key tự động xoá trong khoảng thời gian nhất định.

- Hỗ trợ nhiều loại kiểu dữ liệu khác nhau.

- Hỗ trợ Queue (hàng đợi) thông qua cơ chế PUB/SUB, chúng ta có thể dùng Redis để làm hệ thống hàng đợi cho website xử lý tuần tự từng yêu cầu.

Các kiểu dữ liệu trong Redis:

- **STRING**: string, integer hoặc float. Redis có thể làm việc với cả string, từng phần của string, cũng như tăng/giảm giá trị của integer, float.

- **LIST**: danh sách liên kết của các strings. Redis hỗ trợ các thao tác push, pop từ cả 2 phía của list, trim dựa theo offset, đọc 1 hoặc nhiều items của list, tìm kiếm và xóa giá trị.

- **SET**: tập hợp các string (không được sắp xếp). Redis hỗ trợ các thao tác thêm, đọc, xóa từng phần tử, kiểm tra sự xuất hiện của phần tử trong tập hợp. Ngoài ra Redis còn hỗ trợ các phép toán tập hợp, gồm intersect/union/difference.

- **HASH**: lưu trữ hash table của các cặp khóa-giá trị, trong đó key được sắp xếp ngẫu nhiên, không theo thứ tự nào cả. Redis hỗ trợ các thao tác thêm, đọc, xóa từng phần tử, cũng như đọc tất cả giá trị.

- **ZSET**: là 1 danh sách, trong đó mỗi phần tử là map của 1 chuỗi và 1 kiểu thập phân, danh sách được sắp xếp theo điểm này. Redis hỗ trợ thao tác thêm, đọc, xóa từng phần tử, lấy ra các phần tử dựa theo khoảng của điểm hoặc của chuỗi.

2.2.4. Hệ quản trị cơ sở dữ liệu OrientDB

OrientDB là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở được viết bằng Java. Tuy nhiên, trong OrientDB, các mối quan hệ được xử lý bằng hệ quản trị cơ sở dữ liệu đồ thị kết nối trực tiếp giữa các bản ghi. OrientDB có hỗ trợ cho các lược đồ đầy đủ, lược đồ ít hơn và các mô hình lược đồ hỗn hợp. OrientDB sử dụng thuật toán lập chỉ mục mới có tên MVRB-Tree, xuất phát từ Cây Đỏ-Đen và từ Cây B +; điều này được báo cáo có lợi ích của việc có cả chèn nhanh và tra cứu nhanh.

Các tính năng của nó là:

- Giao dịch: hỗ trợ Giao dịch ACID. Khi gặp sự cố, nó phục hồi các tài liệu đang chờ xử lý.

- GraphDB: quản lý riêng các biểu đồ. Tuân thủ 100% với tiêu chuẩn TinkerPop Blueprints cho hệ quản trị cơ sở dữ liệu đồ thị.

- SQL: hỗ trợ ngôn ngữ SQL với các phần mở rộng để xử lý các mối quan hệ mà không cần tham gia SQL, quản lý cây và biểu đồ của các tài liệu được kết nối.

- Web ready: hỗ trợ HTTP, giao thức RESTful và JSON mà không cần sử dụng các thư viện và thành phần của bên thứ 3.

- Chạy ở mọi nơi: công cụ hoàn toàn là Java thuần túy 100%: chạy trên Linux, Windows và bất kỳ hệ thống nào hỗ trợ công nghệ Java.

Như vậy trong chương này tôi giới thiệu các đặc điểm của bốn loại hệ quản trị cơ sở dữ liệu NoSQL phổ biến là loại cơ sở dữ liệu tài liệu, khóa – giá trị, cột và đồ thị. Mỗi hệ quản trị cơ sở dữ liệu NoSQL cũng có những đặc điểm riêng, và vì thế thường được dùng cho những dự án khác nhau như: MongoDB và Redis là những lựa chọn lưu trữ dữ liệu thống kê ít được đọc và được viết thường xuyên hay Cassandra và OrientDB thường được sử dụng trong môi trường có tính sẵn sàng cao. Luận văn cũng giới thiệu những đặc tính kỹ thuật cơ bản của một số hệ quản trị cơ sở dữ liệu NoSQL phổ biến hiện nay như MongoDB, Redis, Cassandra, OrientDB.

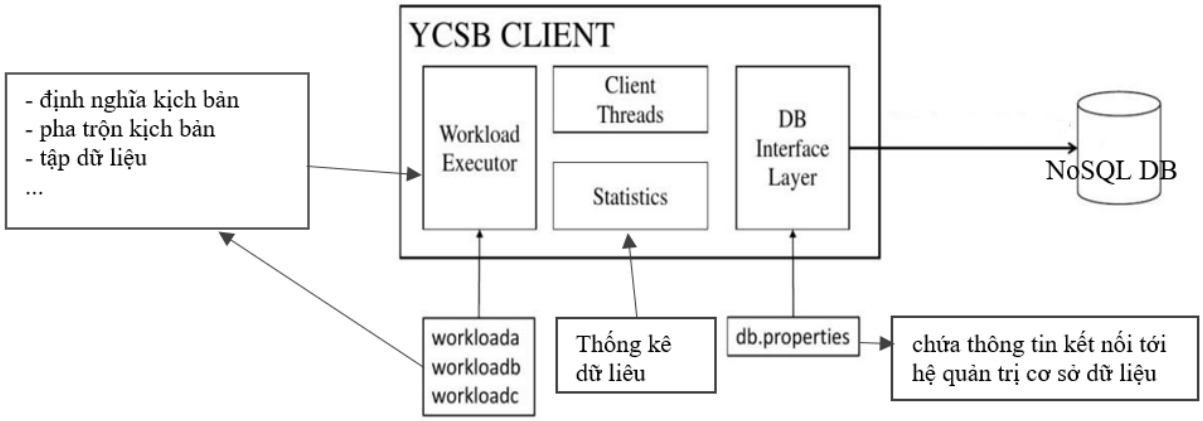
Chương 3. Đánh giá hiệu năng của CSDL NoSQL

Trong phần này, tôi đưa ra các đánh giá thử nghiệm của bốn hệ thống NoSQL bằng cách sử dụng các kịch bản đánh giá hiệu năng công việc được tùy chỉnh khác nhau. Tôi so sánh thời gian chạy và tốc độ xử lý của các hệ thống NoSQL bằng cách thay đổi tỷ lệ của các hoạt động như đọc-chèn, đọc-cập nhật, và quét-chèn. Tỷ lệ của các hoạt động này được thay đổi ở mức 10%, 50% và 90%.

3.1. Cài đặt môi trường

3.1.1. Tổng quan về Yahoo! Cloud Serving Benchmark (YCSB)

Yahoo! Cloud Serving Benchmark (YCSB) là một framework mã nguồn mở để đánh giá và so sánh hiệu năng các hệ thống dữ liệu lớn. YCSB hỗ trợ nhiều loại database khác nhau : Apache HBase, Apache Cassandra, Redis, MongoDB vàVoldemort,…



Hình 3.1.1 Kiến trúc YCSB [17]

Công cụ này bao gồm 2 phần:

- YSCB Client: một bộ tạo workload mở rộng
- Core workload: một tập các kịch bản workload có sẵn

Core workload cung cấp một bức tranh tổng thể về hiệu năng của một hệ thống và YSCB Client cho phép tester khai báo các kịch bản kiểm thử đặc thù với từng hệ thống khác nhau. Đặc biệt các kịch bản này có thể được mở rộng để có thể đo hiệu năng của nhiều loại database khác nhau. YSCB có các lib giao tiếp với nhiều loại database phổ biến bao gồm: Redis, Cassandra, Apache Accumulo, MongoDB, và OrientDB. Ngoài ra, tester có thể bổ sung thêm loại database khác bằng cách chủ động tạo ra các thư viện riêng. Để kiểm thử hiệu năng nhiều

loại dữ liệu khác nhau để so sánh, tester có thể cài đặt nhiều hệ quản trị cơ sở dữ liệu trong cùng một lần triển khai. Sau đó có thể cho chạy các nghiệp vụ lần lượt trên từng loại database như Redis, Cassandra, MongoDB, và OrientDB để đánh giá.

3.1.2. Thông số kỹ thuật

Cấu hình máy tính:

Node	Windows 10 PC
Memory	8G
Processor	Intel Core i7 2600M 3.40GHz
Operating System	Windows 10 enterprise
Disk Type	HDD

Bảng 4.1: Bảng thông số cấu hình máy tính

Các phiên bản của hệ quản trị cơ sở dữ liệu NoSQL

Database Management System	Version
Cassandra	3.0
MongoDB	4.0
OrientDB	3.0.17
Redis	4.0.14

Bảng 4.2: Bảng thông số phiên bản NoSQL

3.1.3. Định nghĩa kịch bản kiểm thử

Để có đưa ra các kịch bản phân tích thử nghiệm, tôi sử dụng công cụ YCSB cho phép đánh giá và so sánh hiệu suất của hệ quản trị cơ sở dữ liệu NoSQL[17][18]. Công cụ đánh giá hiệu năng gồm hai phần: bộ tạo dữ liệu và một tập các kịch bản kiểm tra hiệu suất bao gồm các hoạt động đọc, chèn, cập nhật, quét. Mỗi kịch bản thử nghiệm được gọi là một khối lượng công việc được xác định bởi một tập các tính năng bao gồm các: phần trăm hoạt động đọc, cập nhât, chèn và tổng số hoạt động hoặc số lượng bản ghi được sử dụng. YCSB

cũng cung cấp một tập các khối lượng công việc mặc định có thể được thực thi và xác định bởi các phân trăm đọc, cập nhật, quét và chèn. Khối lượng công việc mặc định là đọc (50%, cập nhật 50%), (đọc 95% và cập nhật 5%) , (đọc 95% và chèn 5%), (quét 95% và 5% chèn).. và từ những khối lượng công việc đó chúng ta có thể kết hợp thành những kịch bản kiểm tra hiểu suất dữa trên những tỉ lệ các hoạt động để phù hợp với yêu cầu của bài toán, các tham số có tỉ lệ như sau:

- operationcount - số lượng hoạt động được thực thi.
- readproportion - tỷ lệ các hoạt động đọc (ví dụ: 0,5 có nghĩa là 50% tổng số hoạt động là lần đọc giá trị hệ quản trị cơ sở dữ liệu).
- updateproportion - tỷ lệ các hoạt động cập nhật.
- scanproportion - tỷ lệ các hoạt động quét (đọc bộ sưu tập đầy đủ).
- insertproportion - tỷ lệ các hoạt động chèn.
- readmodifywriteproportion - tỷ lệ các lô hoạt động đọc, cập nhật và chèn.

✓ Đọc và chèn với khối lượng 100%
- operationcount=1000/10000/100000
- readproportion=1.0
- updateproportion=0
- scanproportion=0
- insertproportion=1.0

✓ Đọc và chèn với khối lượng công việc khác nhau
- operationcount=1000/10000/100000
- readproportion=0.1
- updateproportion=0
- scanproportion=0
- insertproportion=0.9

✓ Đọc và cập nhật với khối lượng công việc khác nhau
- operationcount=1000/10000/100000
- readproportion=0.5
- updateproportion=0.5
- scanproportion=0
- insertproportion=0

✓ Quét và chèn với khối lượng công việc khác nhau

- operationcount=1000/10000/100000
- readproportion=0
- updateproportion=0
- scanproportion=0.5
- insertproportion=0.5

✓ Kết quả khi chạy kịch bản mẫu

```
[OVERALL], RunTime(ms), 393939
[OVERALL], Throughput(ops/sec), 2538.4640769256152
[TOTAL_GCS_PS_Scavenge], Count, 480
[TOTAL_GC_TIME_PS_Scavenge], Time(ms), 827
[TOTAL_GC_TIME_%_PS_Scavenge], Time(%), 0.2099309791617484
[TOTAL_GCS_PS_MarkSweep], Count, 0
[TOTAL_GC_TIME_PS_MarkSweep], Time(ms), 0
[TOTAL_GC_TIME_%_PS_MarkSweep], Time(%), 0.0
[TOTAL_GCs], Count, 480
[TOTAL_GC_TIME], Time(ms), 827
[TOTAL_GC_TIME_%], Time(%), 0.2099309791617484
[CLEANUP], Operations, 1
[CLEANUP], AverageLatency(us), 5650.0
[CLEANUP], MinLatency(us), 5648
[CLEANUP], MaxLatency(us), 5651
[CLEANUP], 95thPercentileLatency(us), 5651
[CLEANUP], 99thPercentileLatency(us), 5651
[INSERT], Operations, 1000000
[INSERT], AverageLatency(us), 385.169593
[INSERT], MinLatency(us), 226
[INSERT], MaxLatency(us), 605183
[INSERT], 95thPercentileLatency(us), 588
[INSERT], 99thPercentileLatency(us), 813
[INSERT], Return=OK, 1000000
C:\YCSB\mongodb\bin>DinhNV
```

Hình 3.2 Kết quả chạy thử ngiệm

3.2. Hiệu suất của các hệ thống NoSQL

Trước hết, luận văn đánh giá hiệu suất của các hệ thống NoSQL riêng lẻ bằng cách tùy chỉnh khối lượng công việc. Tôi đã thay đổi tỷ lệ read, update, insert, scan and read-write-modify của khối lượng công việc và thay đổi số lượng hoạt động và độ dài bản ghi, sau đó theo dõi hiệu suất của từng hệ thống NoSQL.

Hiệu suất của của mỗi hệ thống NoSQL sẽ được đánh giá dựa trên 2 yếu tố:

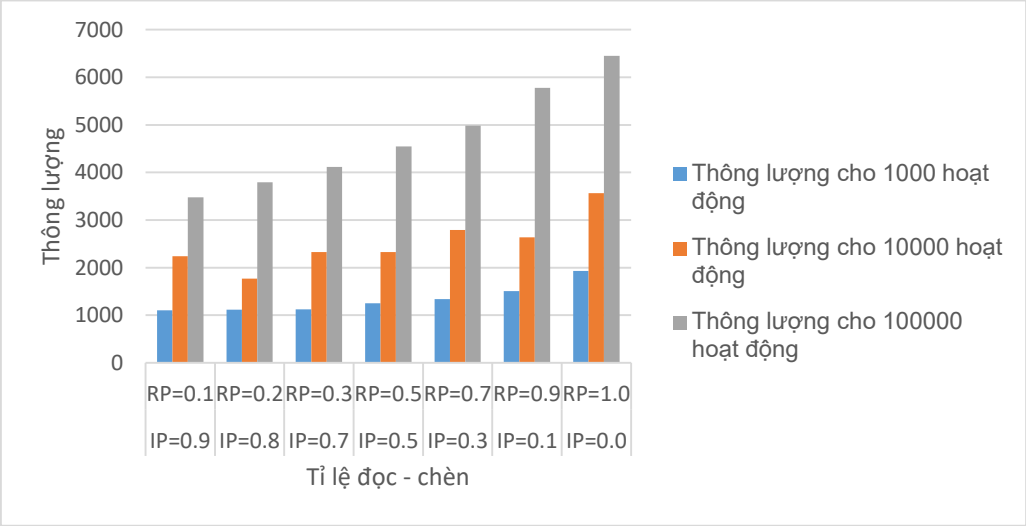
- **Thông lượng (Throughput)**: Thông lượng là một thành phần phi chức năng thuộc danh mục hiệu suất và được đo bằng tổng số giao dịch hoặc

yêu cầu trong một thời gian nhất định hoặc TPS (transaction per second). Nó phản ánh năng lực của máy chủ về khả năng mức độ chịu tải của máy chủ.

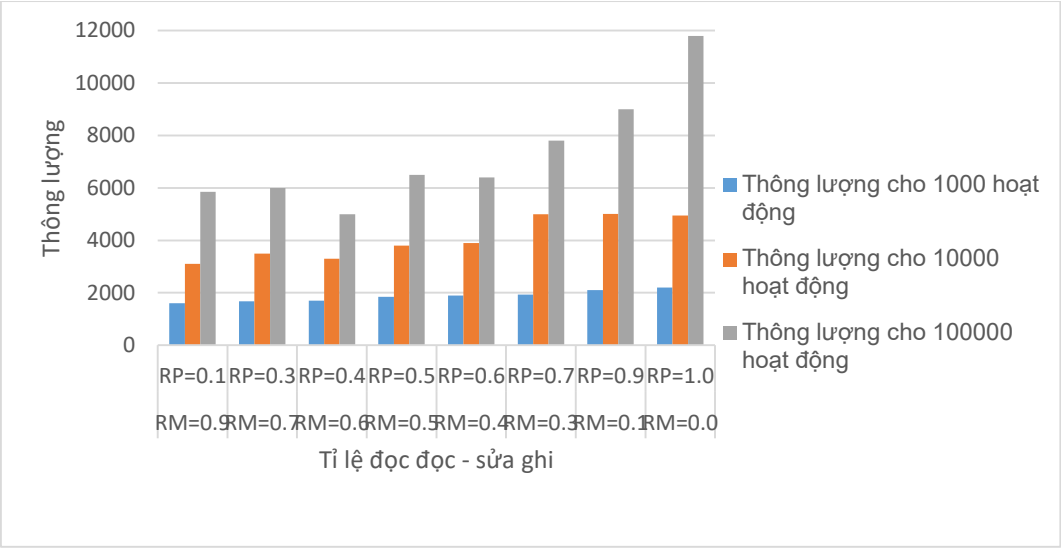
- **Thời gian thực thi (Runtime):** Thời gian thực hiện xong một hoạt động.

3.2.1. MongoDB

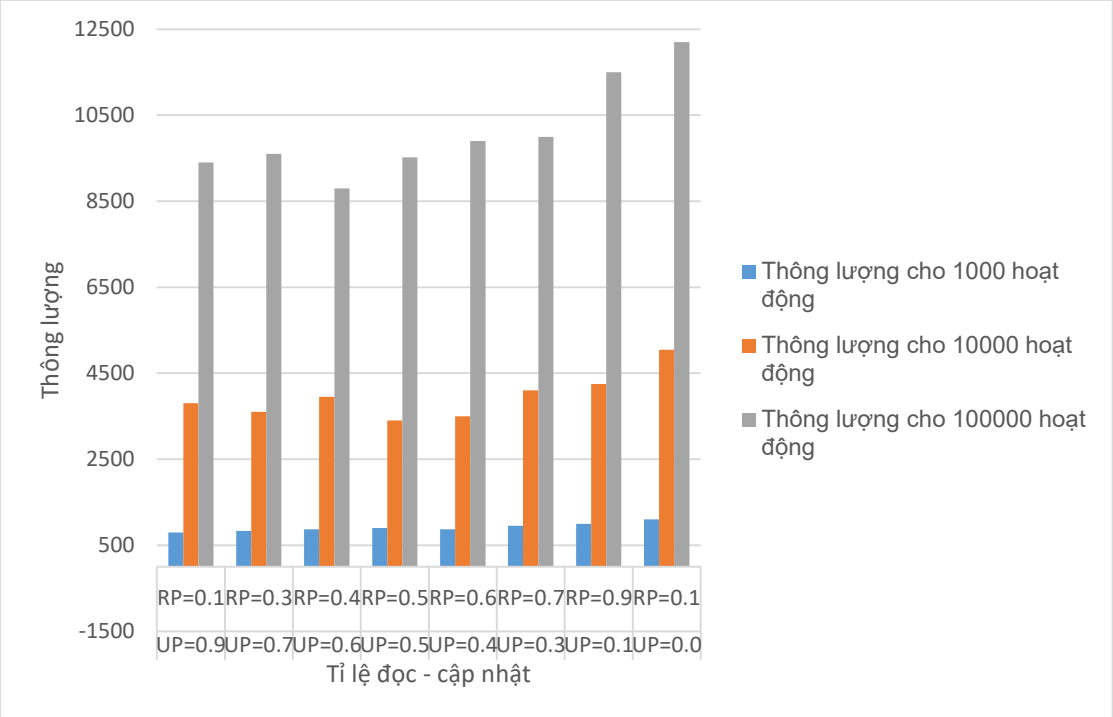
Trước tiên hãy xem hiệu suất của MongoDB. Trong các thử nghiệm, tôi đã thay đổi tỷ lệ của các hoạt động Đọc và các hoạt động Chèn như được thấy trong trục x của Hình 3.2.1. Tôi phát hiện ra rằng khi số lượng hoạt động nhỏ hơn (cho 1000 hoặc 10000), không có nhiều thay đổi trong thông lượng khi tỷ lệ đọc tăng và tỷ lệ chèn giảm đồng thời. Nhưng khi số lượng hoạt động là 100000, có sự gia tăng đáng kể về thông lượng khi tỷ lệ đọc được tăng lên. Vì vậy, tôi đã đi đến kết luận rằng với số lượng hoạt động cao hơn, MongoDB hoạt động tốt hơn với khối lượng công việc nặng hơn. Xu hướng tương tự được thể hiện rõ trong Hình 3.2.1 và 3.2.3. Hình 3.2.2 cho thấy xu hướng thông lượng của hiệu suất của MongoDB khi tỷ lệ đọc được thay đổi theo tỷ lệ đọc-sửa đổi và trong Hình 3.2.3, tôi đã thay đổi tỷ lệ đọc với tỷ lệ cập nhật và thấy rằng xu hướng là tương tự như Hình 3.2.1. Trong Hình 3.2.4, chúng ta thấy rằng hiệu suất giảm đáng kể với sự gia tăng tỷ lệ quét.



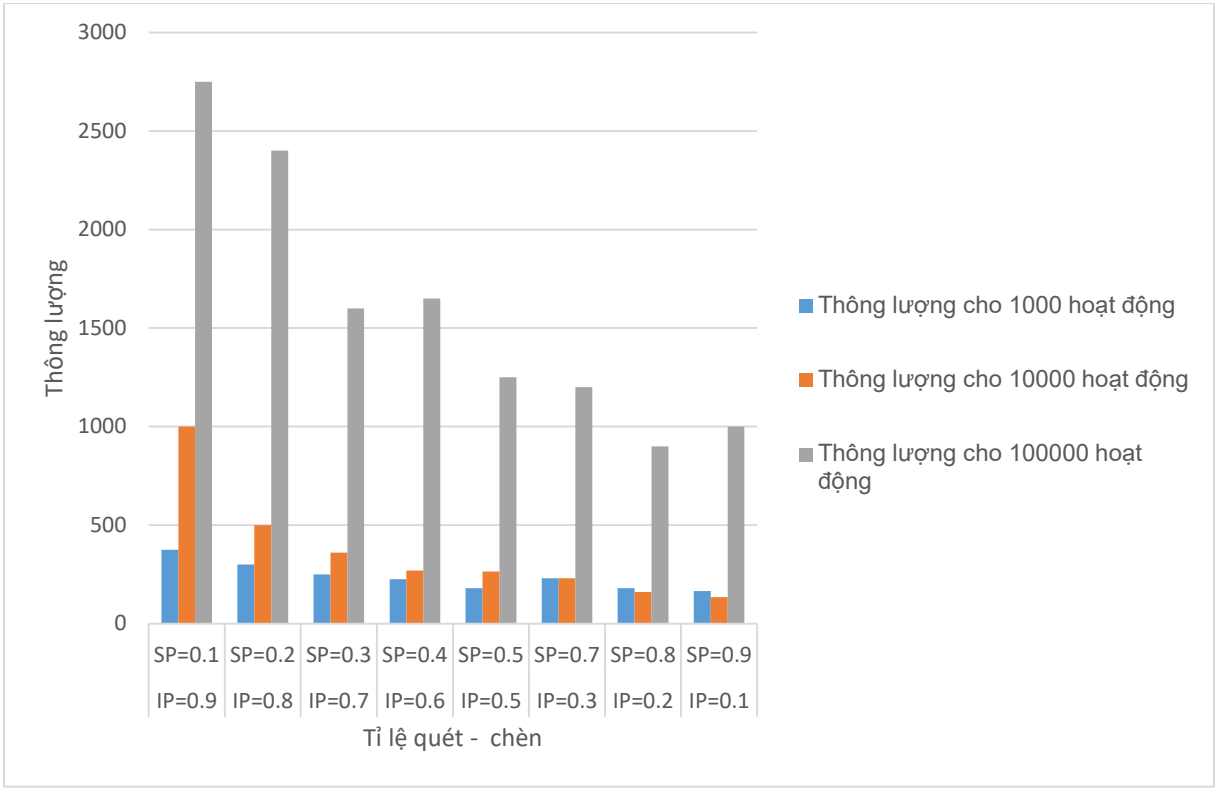
Hình 3.2.1 Kết quả chạy hoạt động đọc và chèn của MongoDB



Hình 3.2.2 Kết quả của hoạt động đọc đọc và sửa ghi của MongoDB



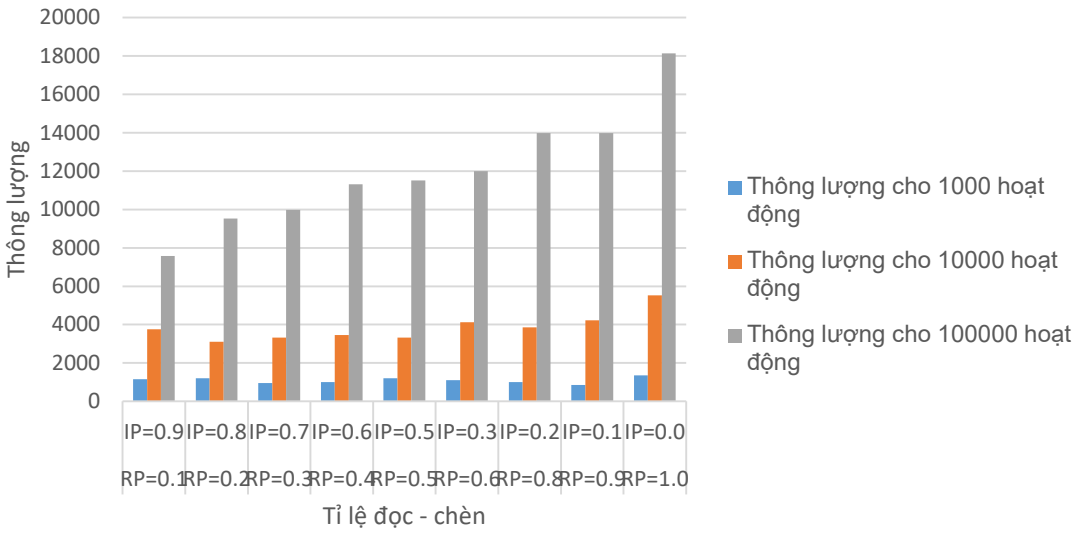
Hình 3.2.3 Kết quả hoạt động đọc – cập nhật MongoDB



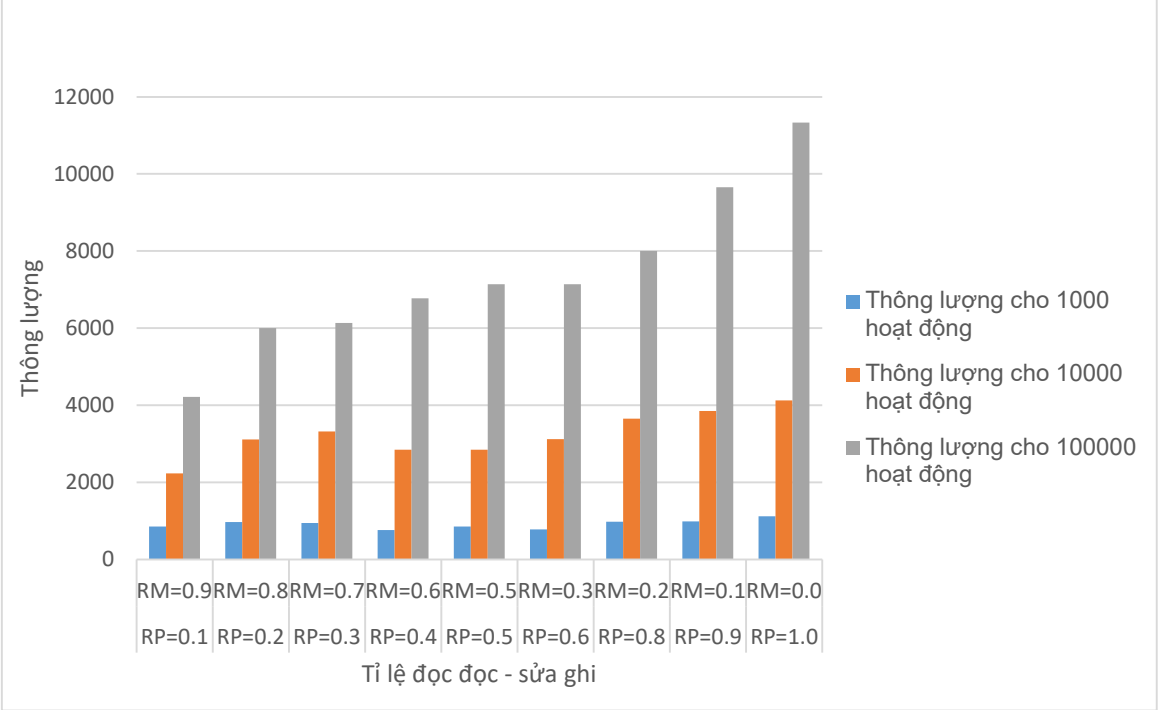
Hình 3.2.4 Kết quả hoạt động quét - chèn MongoDB

3.2.2. OrientDB

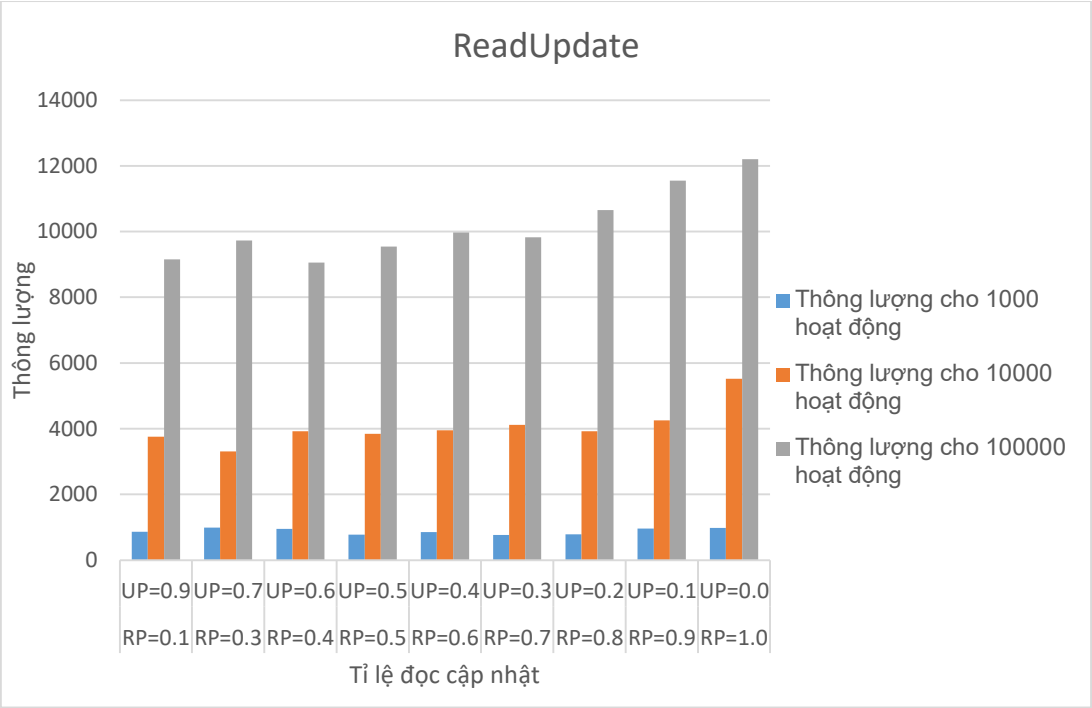
Trong các kịch bản kiểm tra hiệu suất, tôi đã thay đổi tỷ lệ của hoạt động Đọc và hoạt động Chèn giống như cách tôi đã kiểm tra MongoDB. Tôi kết luận rằng thông lượng tăng khi số lượng hoạt động tăng từ 1000 lên 10000 và 100000. Khi tôi tăng tỷ lệ đọc và giảm tỷ lệ chèn, thông lượng của chúng không thay đổi nhiều so với số hoạt động 1000 và 10000. Tuy nhiên, thông lượng tăng cao khi giảm tỉ lệ hoạt động chèn và tăng tỉ lệ hoạt động đọc khi số lượng hoạt động là 100000. Vì vậy, tôi có thể loại trừ rằng thông lượng tăng lên khi số lượng hoạt động tăng lên. Do đó, OrientDB có khả năng mở rộng tốt. Tôi đã quan sát và đi đến kết luận tương tự khi tôi thay đổi tỷ lệ đếm cho Đọc, Cập nhật và Đọc, Đọc-Sửa đổi-Ghi như thể hiện trong Hình 3.2.6 và 3.2.7. Trong Hình 3.2.8, thấy rằng có một sự gia tăng đáng kể về hiệu suất khi gia tăng tỷ lệ quét. Khi số hoạt động là 100000 có thể thấy thông lượng tăng mạnh khi thay đổi tỷ lệ các hoạt động.



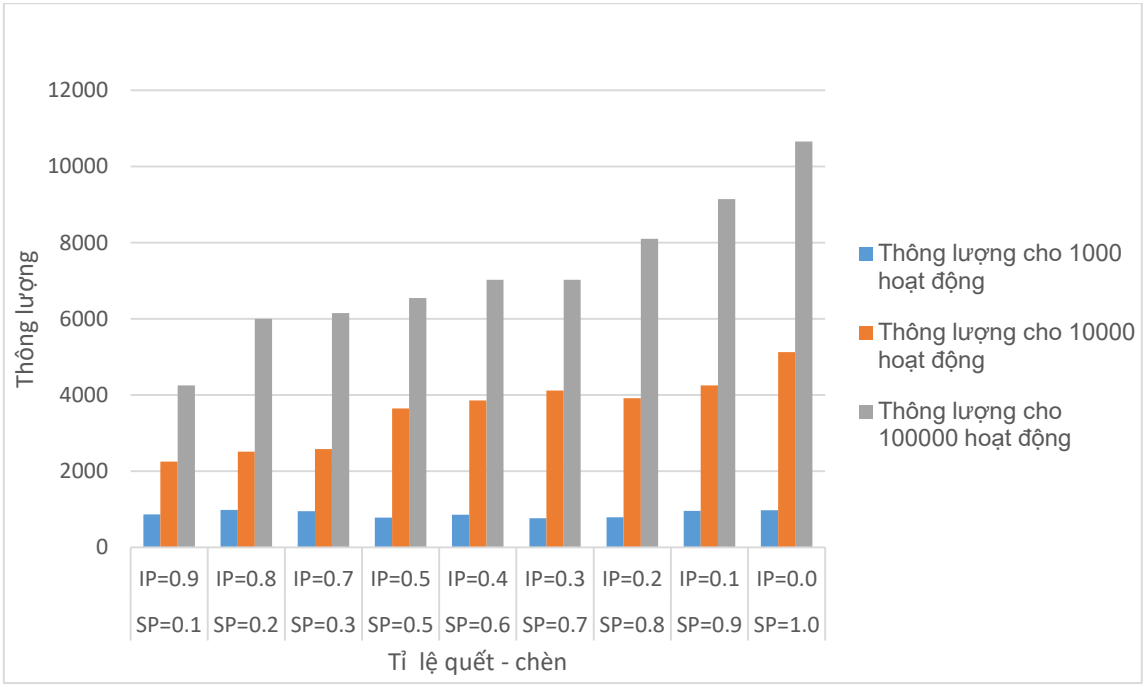
Hình 3.2.5 Kết quả chạy hoạt động đọc và chèn của OrientDB



Hình 3.2.6 Kết quả chạy của hoạt động đọc đọc và sửa ghi của OrientDB



Hình 3.2.7 Kết quả hoạt động đọc và cập nhật OrientDB

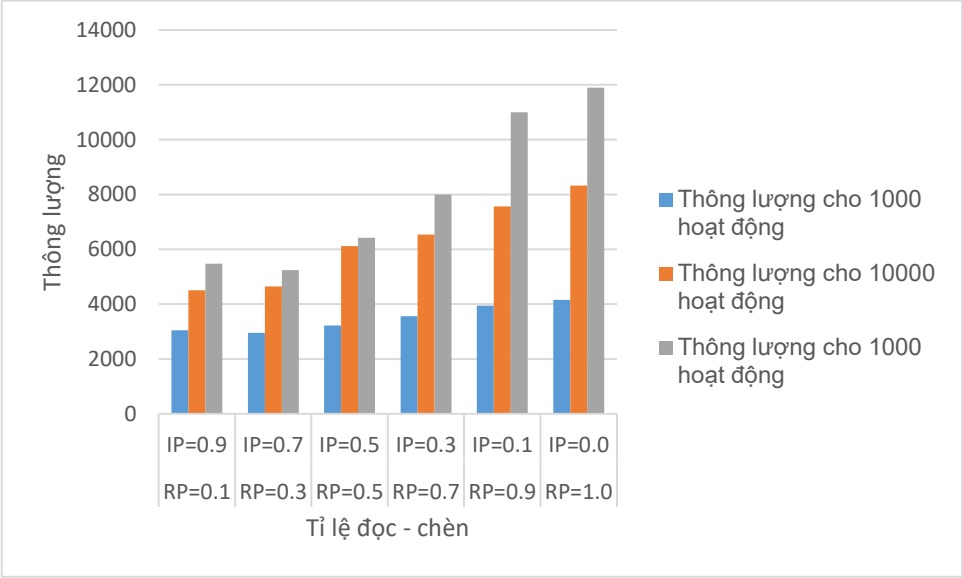


Hình 3.2.8 Kết quả hoạt động quét và chèn OrientDB.

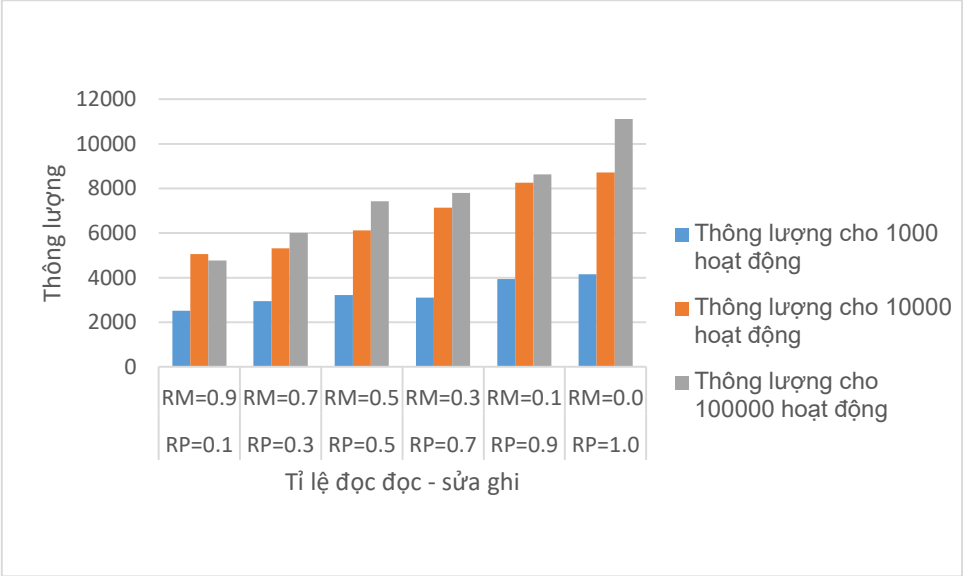
3.2.3. Redis

Hình 3.2.9, 3.2.10, 3.2.11 và 3.2.12 cho thấy Redis thực hiện như thế nào trên các khối lượng công việc khác nhau với số lượng hoạt động tăng dần. Có thể thấy sự gia tăng thông lượng khi tăng Tỷ lệ đọc so với Tỷ lệ chèn ở Hình 3.2.9 và tỷ lệ Đọc-Sửa đổi-Viết trong Hình 3.2.10. Chúng ta cũng có thể nhận

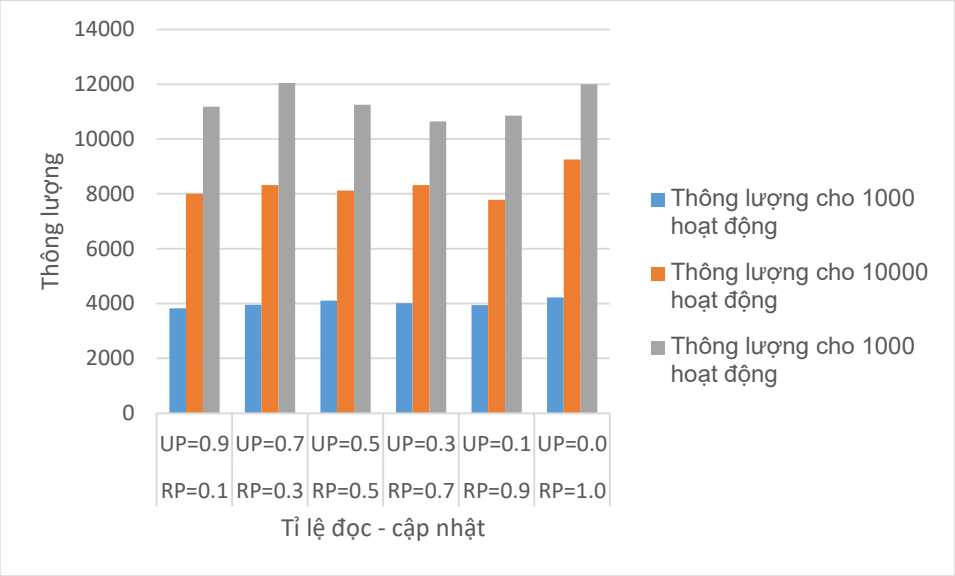
thấy ngay sự khác biệt về hiệu suất của Redis khi so sánh với MongoDB và OrientDB. Trong trường hợp Đọc-Chèn và Đọc-Đọc Sửa đổi Viết, dường như không có nhiều sự khác biệt về thông lượng cho số lượng hoạt động bằng 1000 và 10000. Về việc thay đổi tỷ lệ đọc và cập nhật, có rất ít sự khác biệt về thông lượng khi tỷ lệ đọc được tăng lên và tỷ lệ cập nhật giảm đồng thời như hiển thị trong Hình 3.2.11. Ngoài ra, chúng ta có thể thấy rằng hiệu suất giảm khi tỷ lệ quét được tăng lên và tỷ lệ chèn giảm như chúng ta có thể thấy trong Hình 3.2.12.



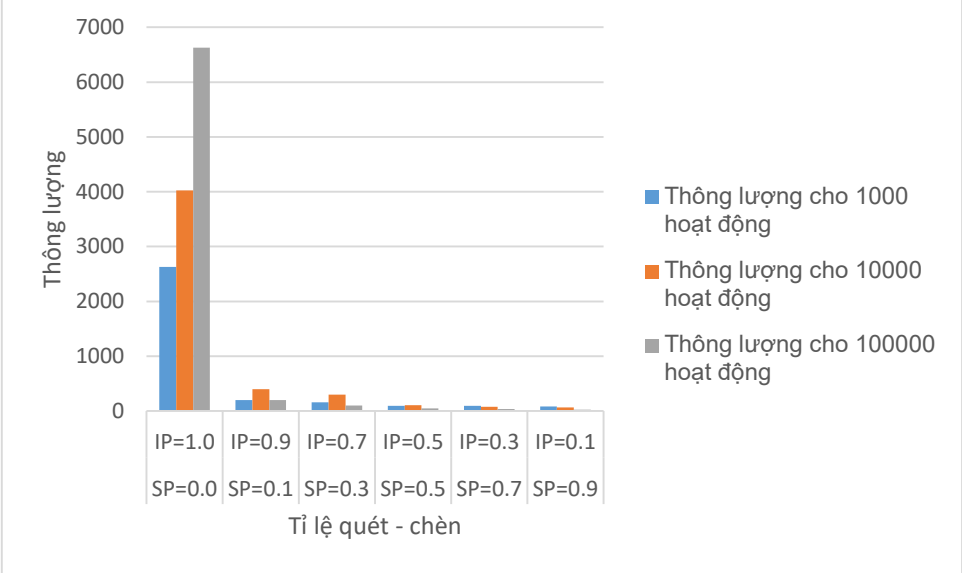
Hình 3.2.9 Kết quả hoạt động đọc và chèn của Redis



Hình 3.2.10 Kết quả hoạt động đọc đọc và sửa ghi của Redis



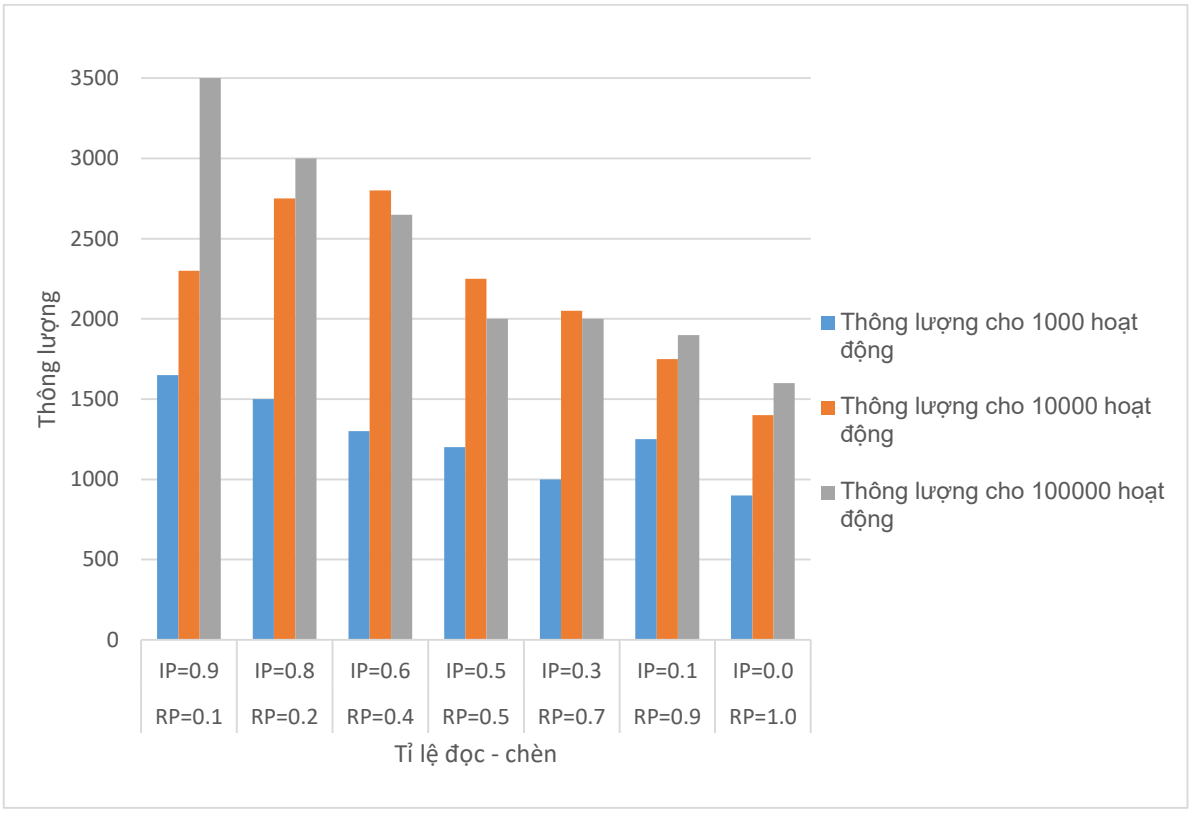
Hình 3.2.11 Kết quả hoạt động đọc và cập nhật của Redis



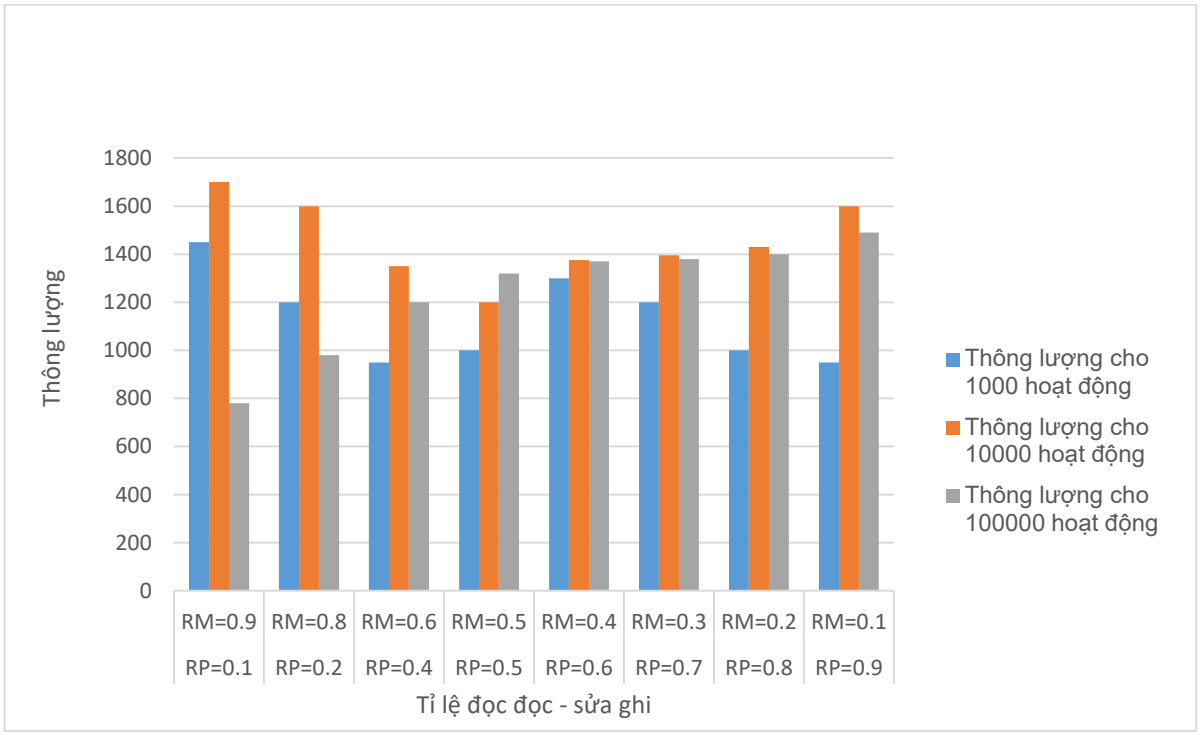
Hình 3.2.12 Kết quả hoạt động quét và chèn của Redis

3.2.4. Cassandra

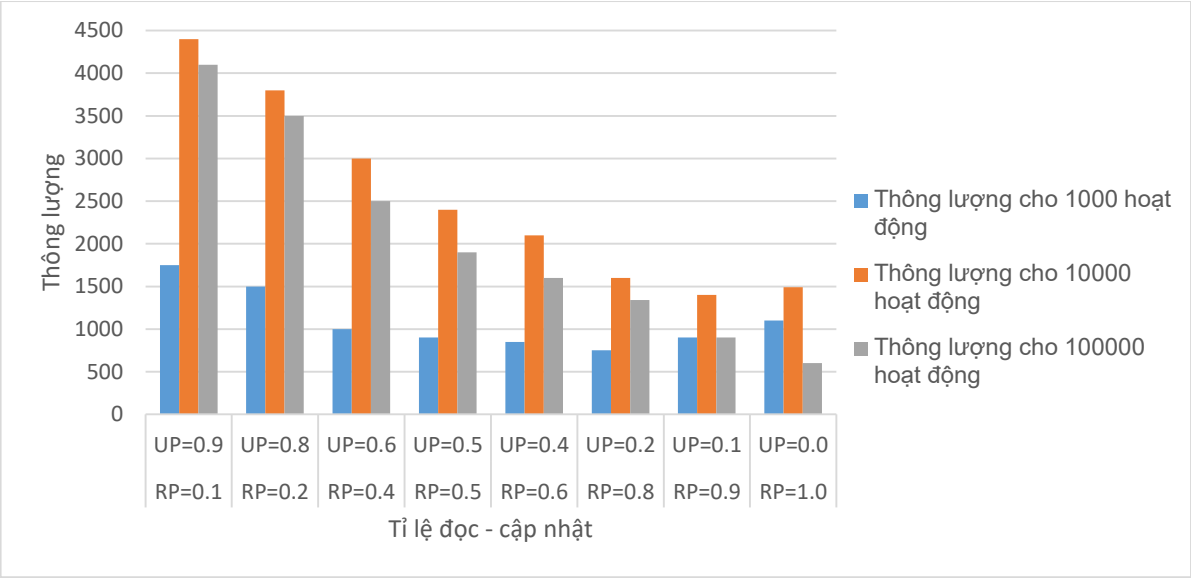
Hiệu suất của Cassandra được thể hiện trong Hình 3.2.13, 3.2.14, 3.2.15 và Hình 3.2.16. Có thể nhận thấy rằng hiệu năng của Cassandra khác so với các hệ thống NoSQL khác. Thông lượng vẫn gần như giống nhau cho số thao tác = 10000 và 100000 khi thay đổi tỷ lệ của các hoạt động cho hoạt động Đọc-Chèn và các hoạt động Đọc-Đọc-Sửa đổi-Viết. Cassandra hoạt động tốt khi số lượng thao tác đọc cao. Đối với các hoạt động Quét-Chèn khi tỷ lệ quét được tăng lên thì thông lượng vẫn giữ nguyên cho số lượng hoạt động cao hơn.



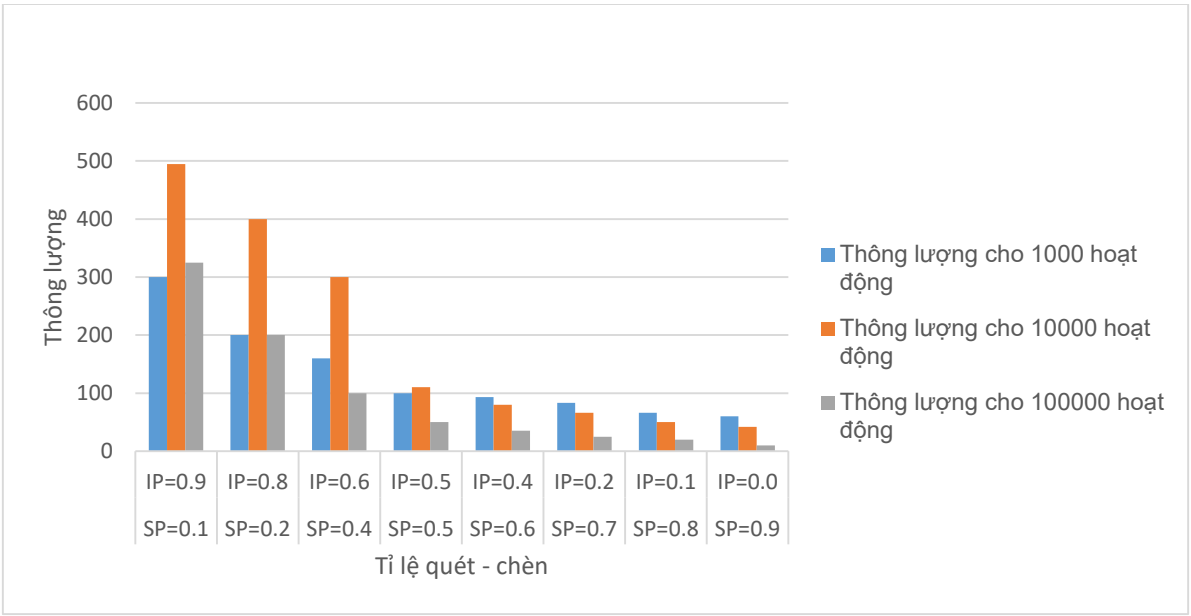
Hình 3.2.13 Kết quả hoạt động đọc và chèn của Cassandra



Hình 3.2.14 Kết quả hoạt động đọc đọc và sửa ghi của Cassandra



Hình 3.2.15 Kết quả hoạt động đọc và cập nhật của Cassandra



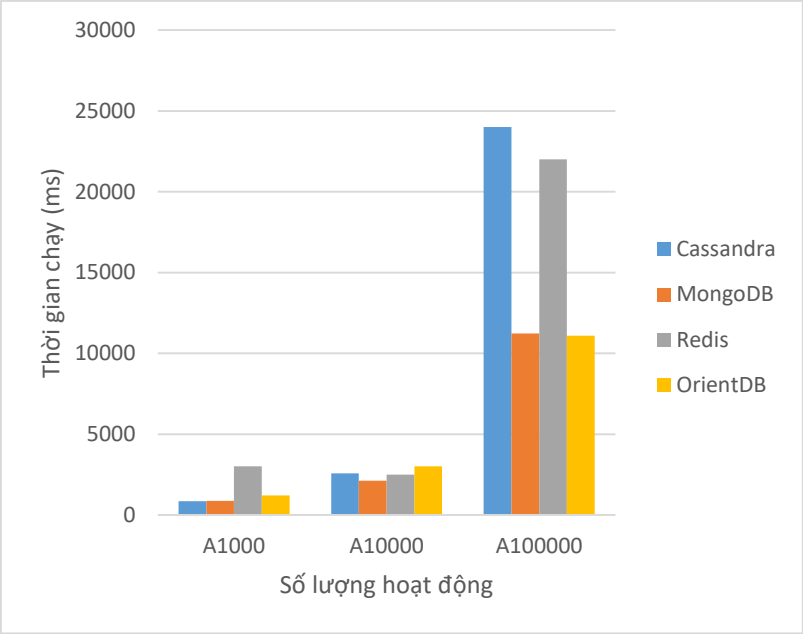
Hình 3.2.16 Kết quả hoạt động quét và chèn của Cassandra

3.3. Đánh giá hiệu suất

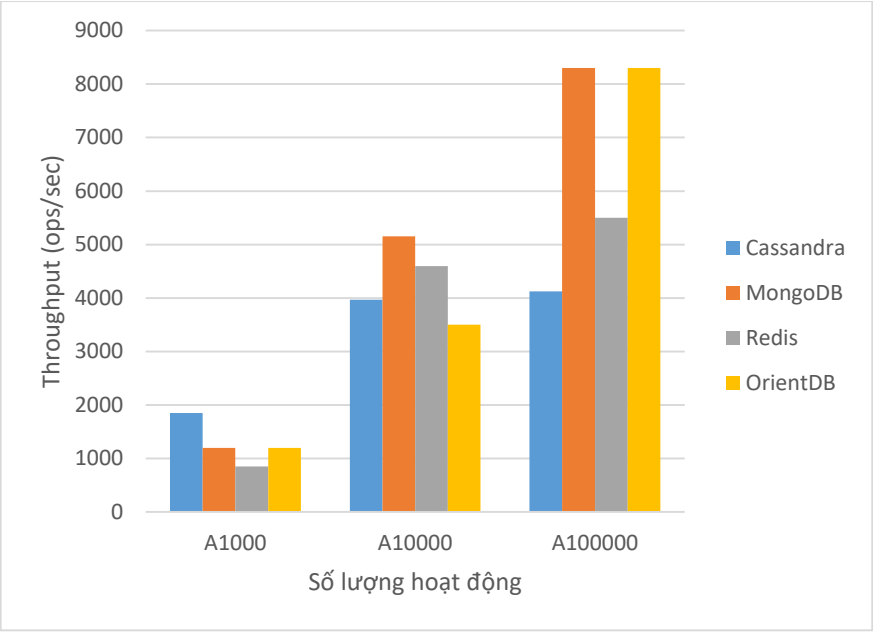
Trong phần này, tôi thực hiện nghiên cứu so sánh các kịch bản đã xây dựng ở trên và đưa ra đánh giá hiệu năng của bốn hệ thống NoSQL bằng cách sử dụng các kịch bản có khối lượng công việc được tùy chỉnh khác nhau. Tôi so sánh thời gian chạy và thông lượng của các hệ thống NoSQL bằng cách thay đổi tỷ lệ của các hoạt động Đọc- Chèn, Đọc-Cập nhật, Đọc-Đọc Sửa đổi Ghi và Quét-Chèn. Tỷ trọng của các hoạt động này thay đổi ở mức 10%, 50%và 90%.

3.3.1 Đọc và chèn với khối lượng 100%

Hình 3.3.1 và 3.3.2 cho thấy hiệu năng của bốn hệ thống NoSQL cho các hoạt động Đọc 100 % và Chèn 100%. Trong Hình 3.3.1 và 3.3.2 có thể thấy rằng khi số lượng thao tác Chèn tăng lên, hiệu suất của Cassandra giảm, nó có thời gian chạy cao nhất và thông lượng thấp nhất. OrientDB và MongoDB thực hiện gần như theo cách tương tự. Chúng cung cấp hiệu suất tốt hơn so với Cassandra hoặc Redis. Vì vậy, tôi có thể kết luận rằng đối với Chèn khối lượng công việc lớn, tốt nhất là chọn MongoDB hoặc OrientDB.

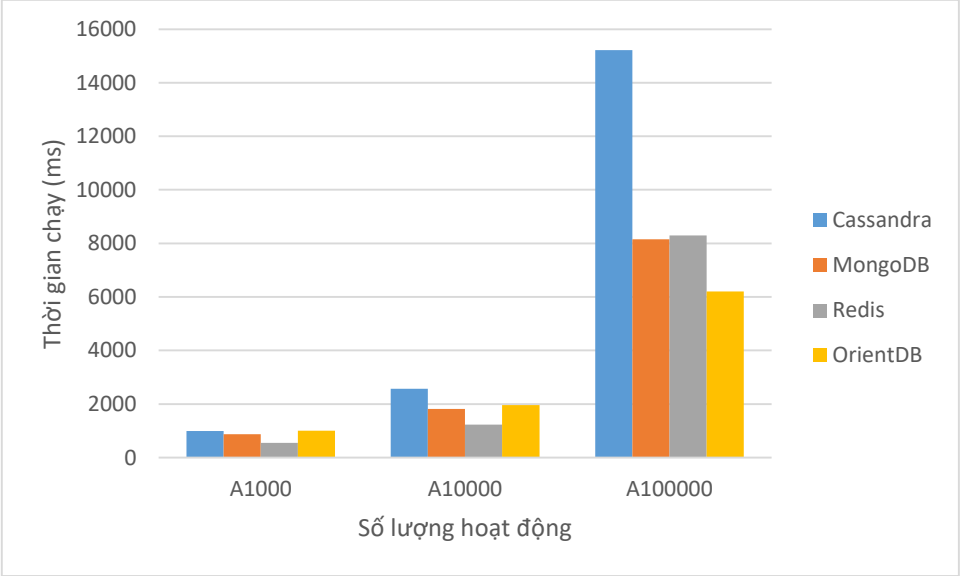


Hình 3.3.1 Kết quả thời gian chèn

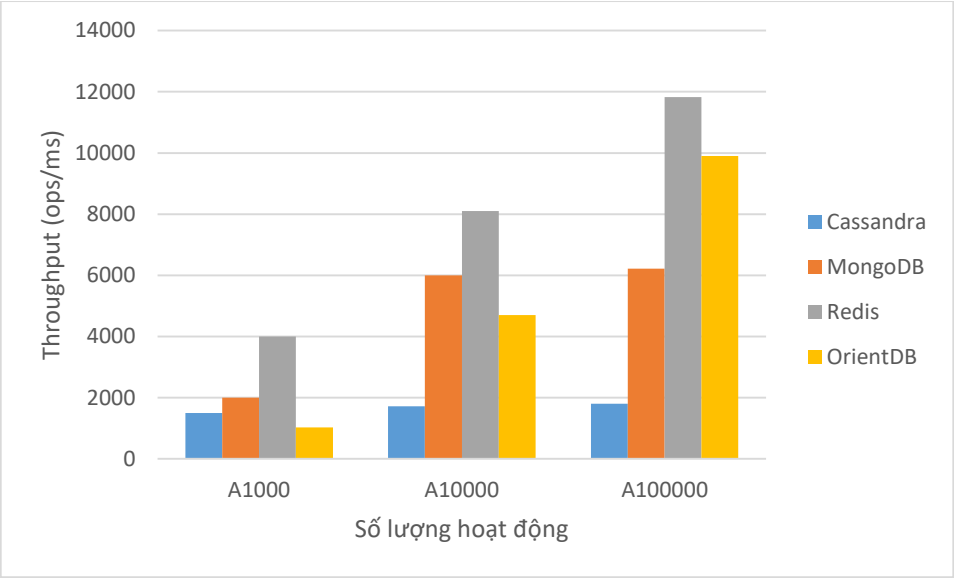


Hình 3.3.2 Thông lượng hoạt động chèn

Từ Hình 3.3.3 và 3.3.4, tôi có thể kết luận rằng đối với các hoạt động Đọc cũng vậy, Cassandra có kết quả thực hiện kém nhất, và có thời gian chạy cao nhất và thông lượng thấp nhất. Redis cung cấp thông lượng cao nhất và mất thời gian ngắn nhất để thực hiện các hoạt động Đọc cho khối lượng công việc cao hơn. Do đó, tôi có thể kết luận rằng đối với Đọc khối lượng công việc lớn, tốt nhất là chọn Redis so với những CSDL NoSQL khác.



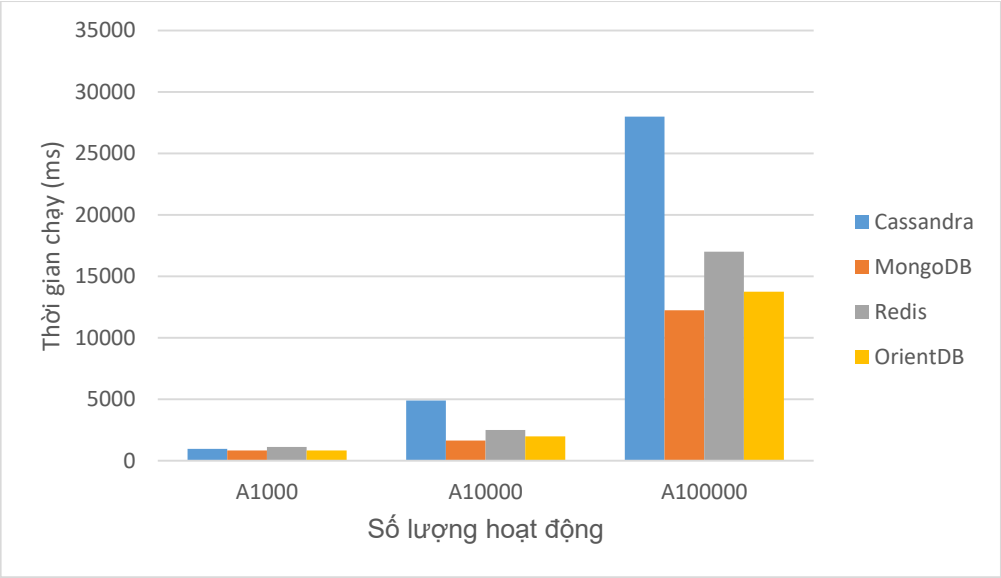
Hình 3.3.3 Kết quả thời gian chạy của hoạt động đọc



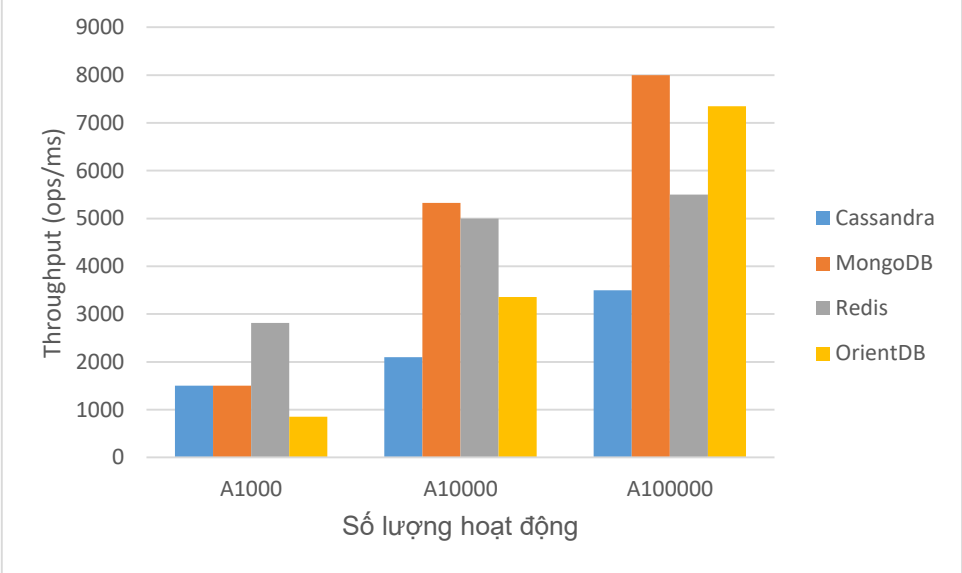
Hình 3.3.4 Kết quả thông lượng của hoạt động đọc

3.3.2 Đọc và chèn với khối lượng công việc khác nhau

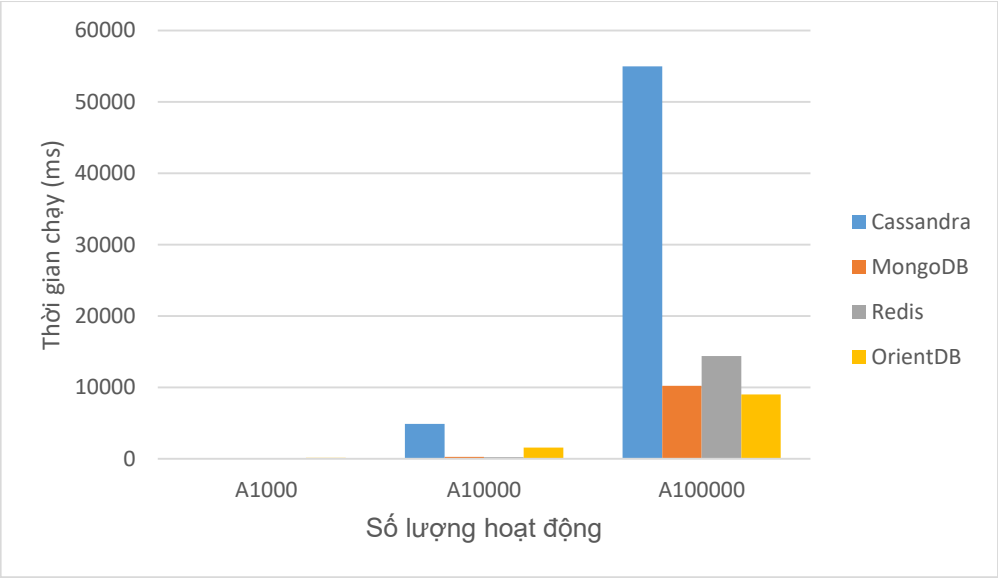
Trong phần này, tôi xem xét hiệu suất của các hệ thống NoSQL bằng cách thay đổi tỷ lệ Đọc và Chèn của các kịch bản đo lường hiệu suất. Trục x hiển thị thông lượng của các hệ quản trị cơ sở dữ liệu NoSQL và trục y hiển thị số lượng hoạt động. Tôi ghi lại thời gian chạy và thông lượng cho các lần chạy vận hành là 1000, 10000 và 100000. Có thể quan sát rằng với tỉ lệ Đọc10% và Chèn 90% MongoDB hoạt động tốt nhất trong các hệ quản trị cơ sở dữ liệu khi thông lượng tỷ lệ Đọc tăng và thời gian xử lý của MongoDB giảm. Hiệu suất của Redis tăng tương đối khi tỉ lệ đọc và chèn hoạt động ổn định. Vì vậy, từ Hình 3.3.5, 3.3.6, 3.3.7, 3.3.8, 3.3.9 và 3.3.10, chúng ta có thể suy luận rằng đối với khối lượng công việc có hoạt động Chèn cao, chúng ta nên chọn MongoDB và cho khối lượng công việc có hoạt động Đọc cao, chúng ta nên chọn Redis. Cassandra kém nhất trong các CSDL vì thời gian thực thi cao và tốc độ xử lý thấp.



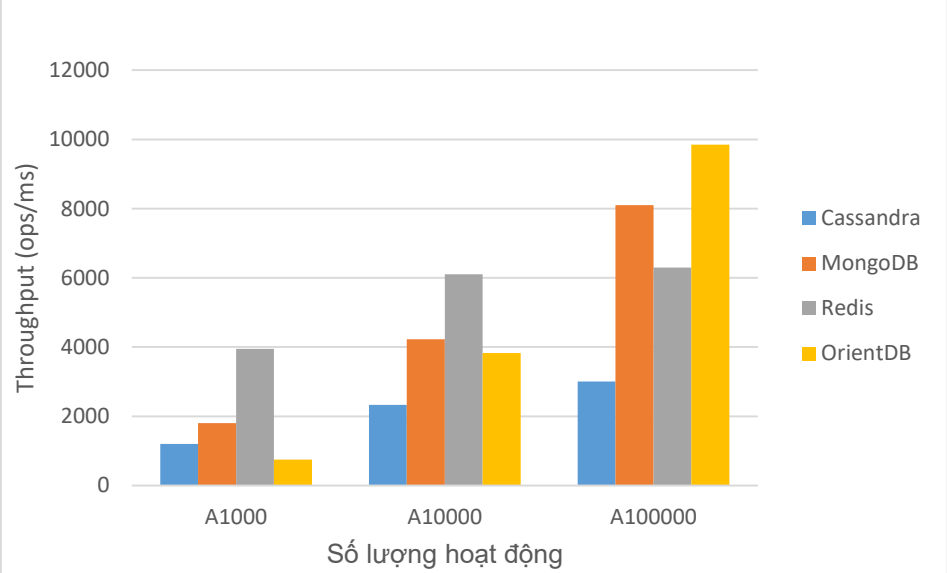
Hình 3.3.5 Kết quả thời gian chạy 10% đọc - 90%-chèn



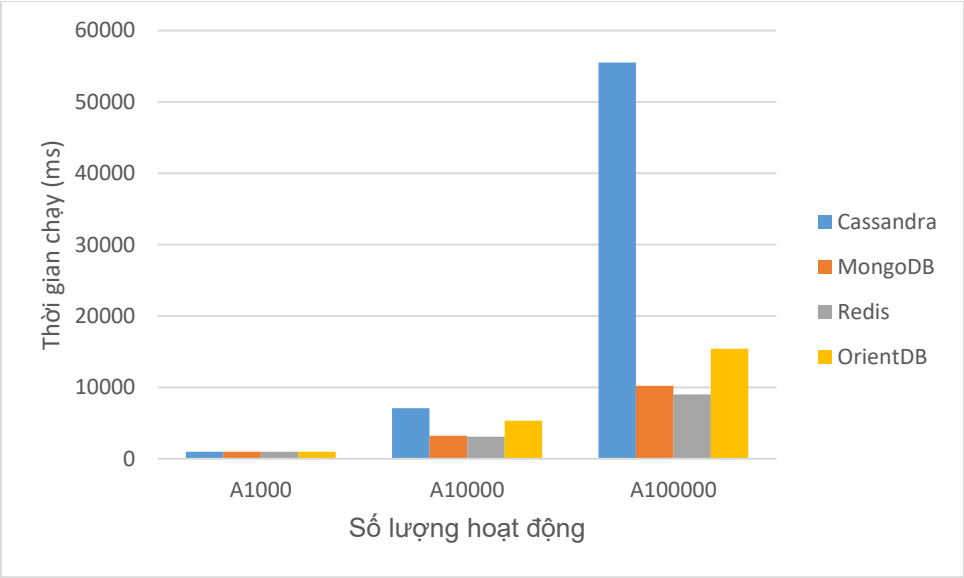
Hình 3.3.6 Kết quả thông lượng 10% đọc - 90% chèn



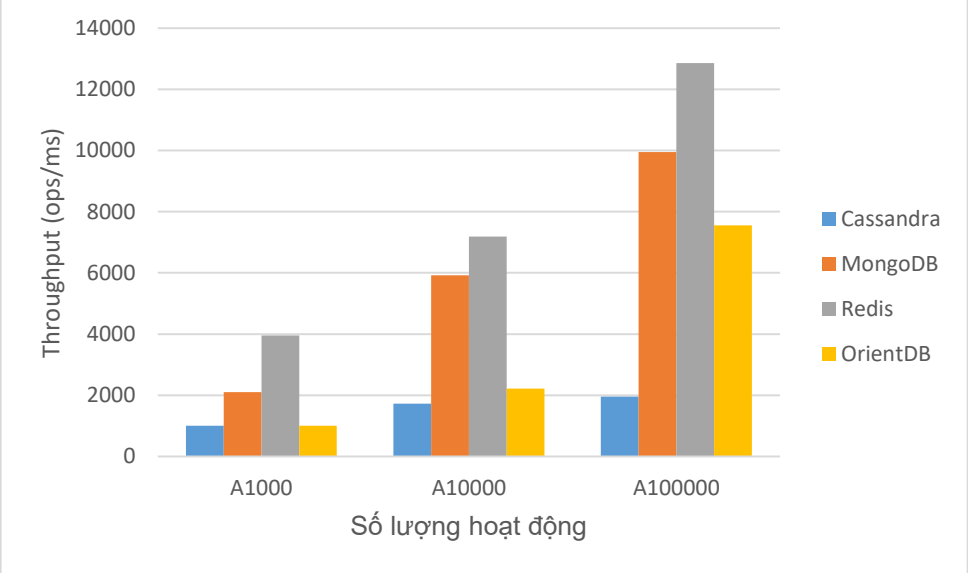
Hình 3.3.7 Kết quả thời gian chạy 50% đọc - 50% chèn



Hình 3.3.8 Kết quả thông lượng 50% đọc - 50% chèn



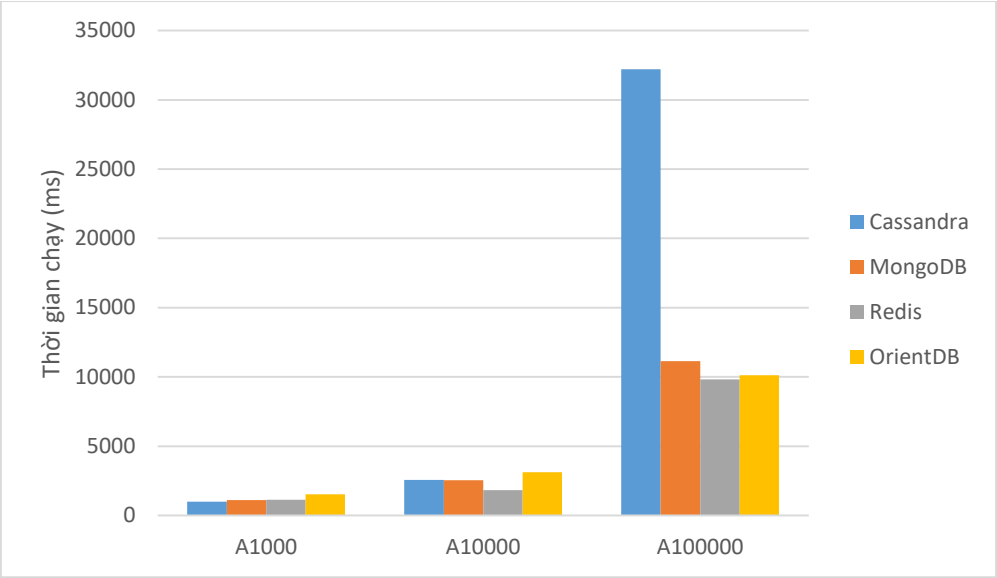
Hình 3.3.9 Kết quả thời gian chạy 90% đọc - 10% chèn



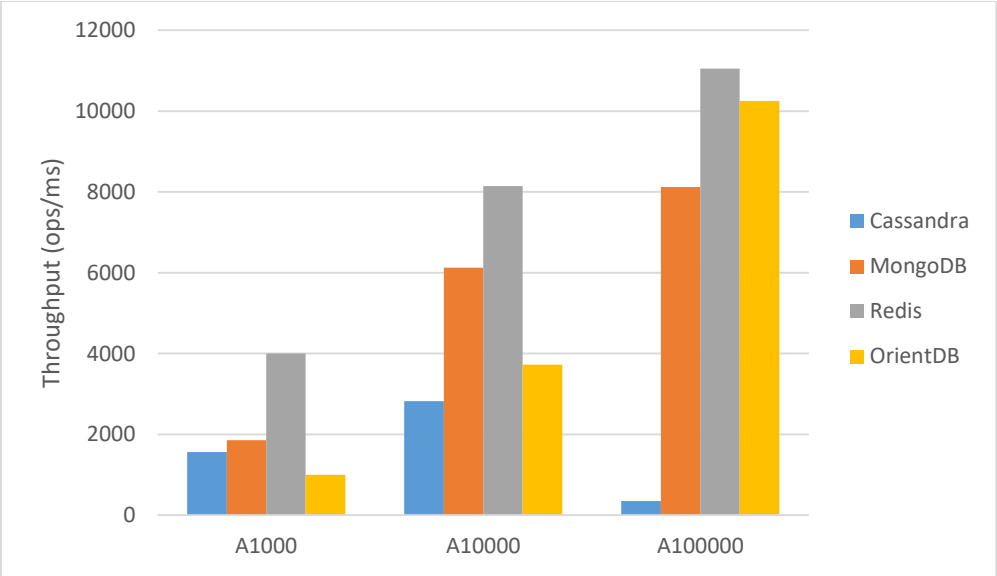
Hình 3.3.10 Kết quả thông lượng 90% đọc - 10% chèn

3.3.3 Đọc và cập nhật với khối lượng công việc khác nhau

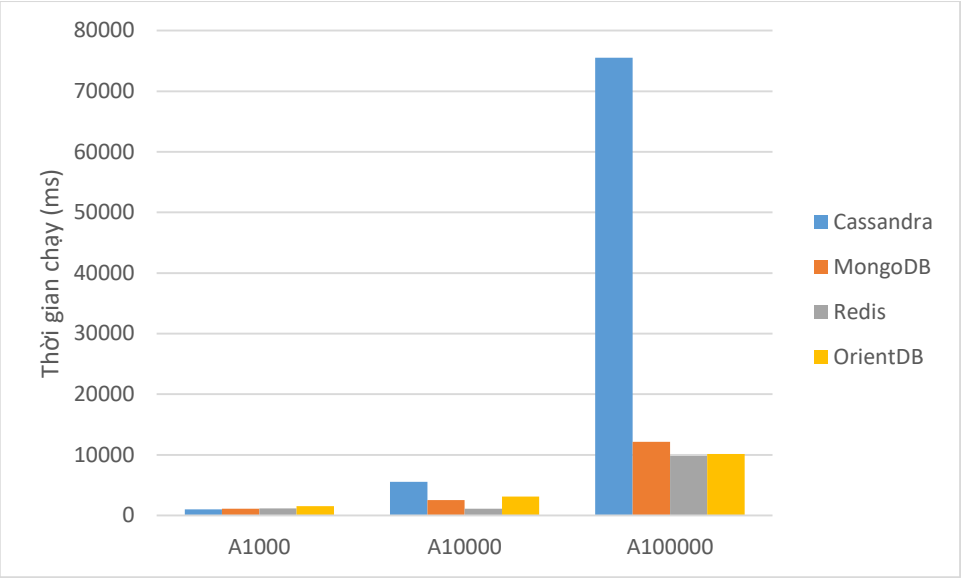
Tôi thay đổi tỷ lệ Đọc và Cập nhật trong phần này. Trục y biểu thị số lượng hoạt động. Tôi ghi lại thời gian chạy và tốc độ xử lý cho số thao tác = 1000, 10000 và 100000 bản ghi. Từ Hình 3.3.11 đến 3.3.14, tôi có thể thấy khi tỷ lệ đọc và cập nhật thay đổi, hiệu suất của các hệ thống NoSQL thay đổi.



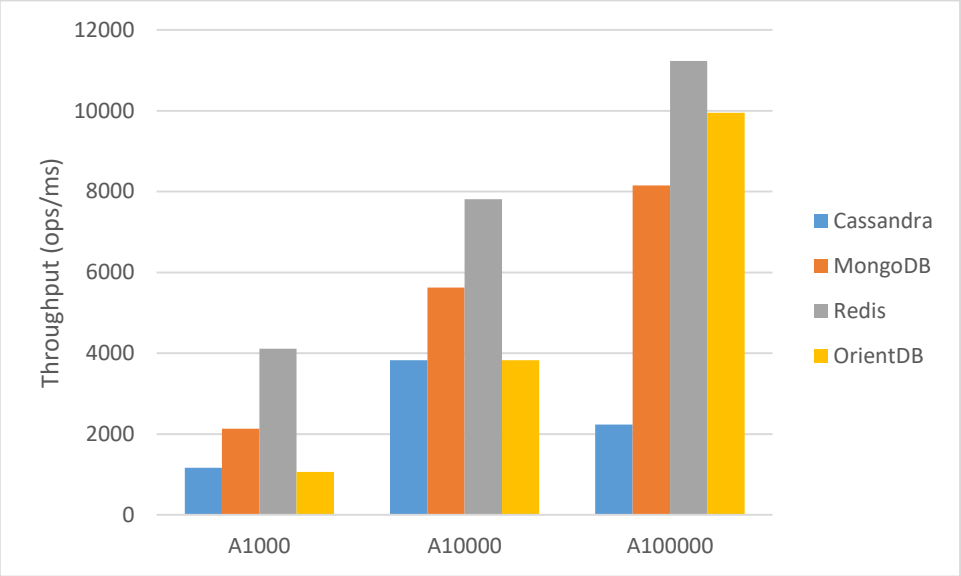
Hình 3.3.11 Kết quả thời gian chạy 10% đọc và 90% cập nhật



Hình 3.3.12 Kết quả thông lượng 10% đọc - 90% cập nhật



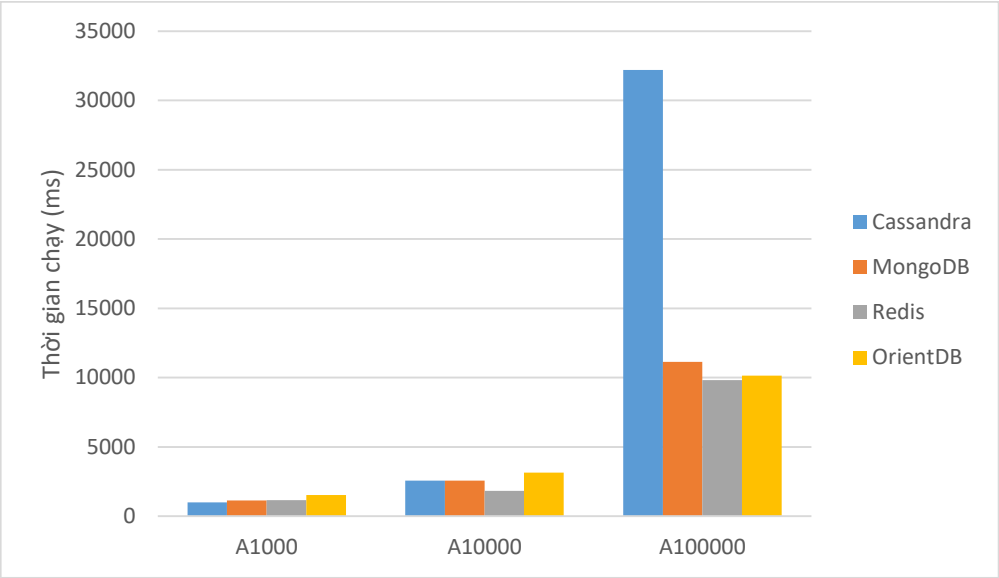
Hình 3.3.13 Thơi gian chạy 50% đọc - 50% cập nhật



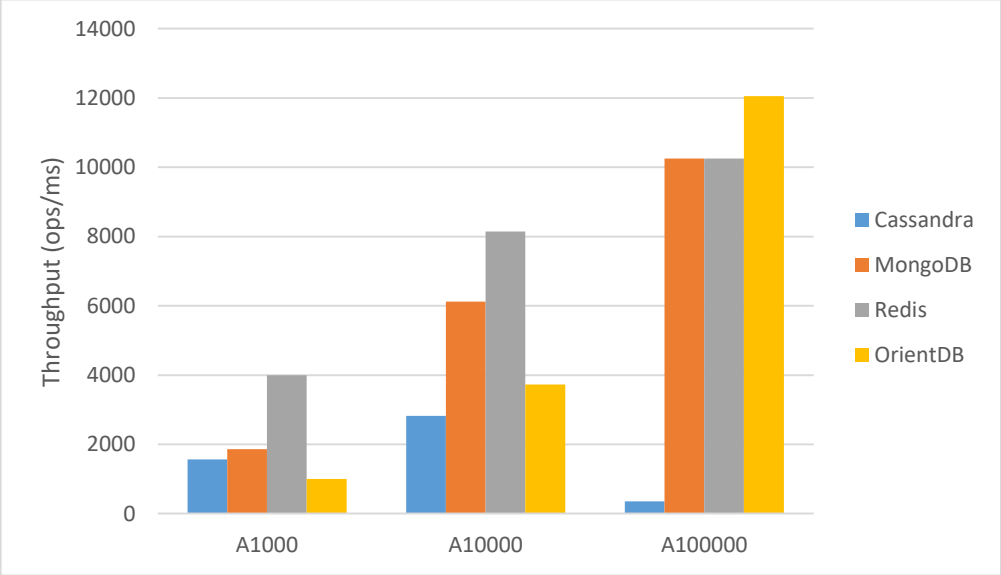
Hình 3.3.14 thông lượng 50% đọc - 50% cập nhật

Đối với khối lượng công việc lớn, Redis thực hiện tốt nhất và khi tỷ lệ cập nhật tăng lên, hiệu suất của MongoDB và OrientDB được cải thiện và như chúng ta thấy trong Hình 3.3.15 và 3.3.16, hiệu suất của OrientDB là tốt nhất khi tỷ lệ cập nhật là 90%. Vì vậy, đối với khối lượng công việc có kết hợp các hoạt động Đọc và Cập nhật, Redis được khuyến khích cho tỷ lệ cập nhật thấp hơn và cho tỷ lệ cập nhật cao có thể chọn OrientDB hoặc MongoDB. Cassandra mất

nhiều thời gian nhất để chạy và thông lượng của nó là thấp nhất cũng như cho tất cả các tỷ lệ của hoạt động đọc và cập nhật.



Hình 3.3.15 Thời gian chạy 10% đọc - 90% cập nhật

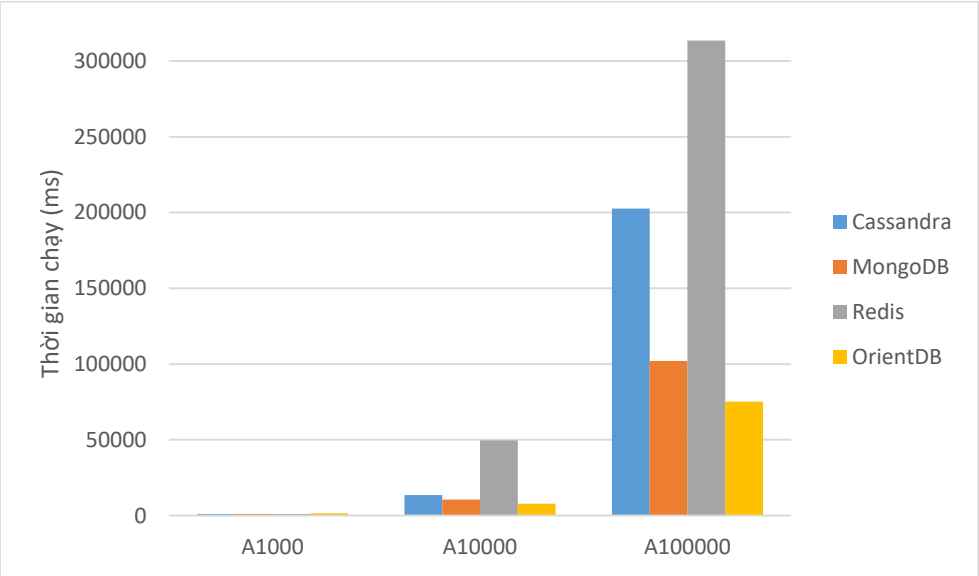


Hình 3.3.16 Thông lượng 10% đọc - 90% cập nhật

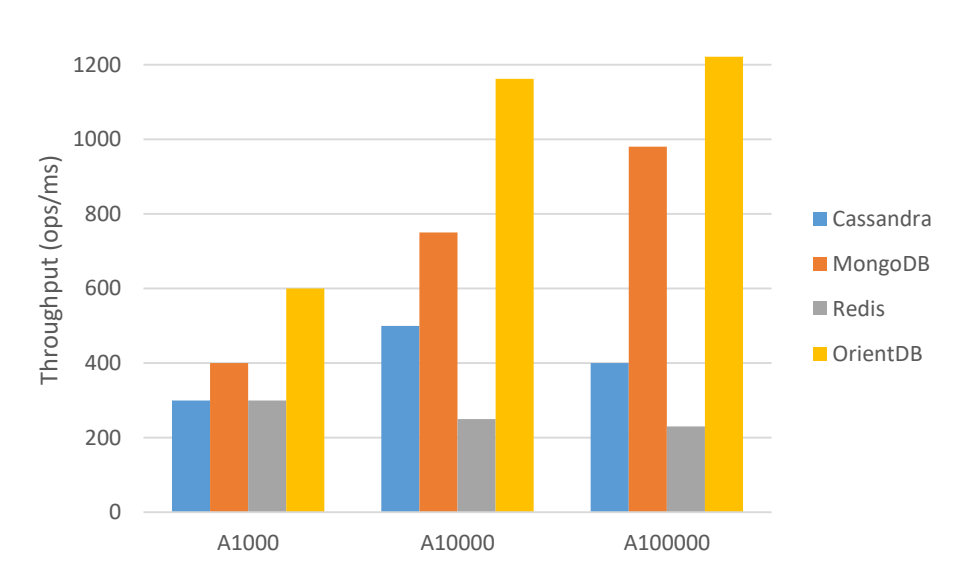
3.3.4 Quét và chèn với khối lượng công việc khác nhau

Hiệu suất của bốn hệ thống NoSQL khi chúng tôi thay đổi tỷ lệ Quét và Cập nhật và tăng số lượng hoạt động được báo cáo trong Hình 3.3.17 đến 3.3.22. Trái ngược hoàn toàn với khối lượng công việc có sự kết hợp của các hoạt động đọc và chèn hoặc các hoạt động đọc và cập nhật, chúng ta có thể quan sát rằng

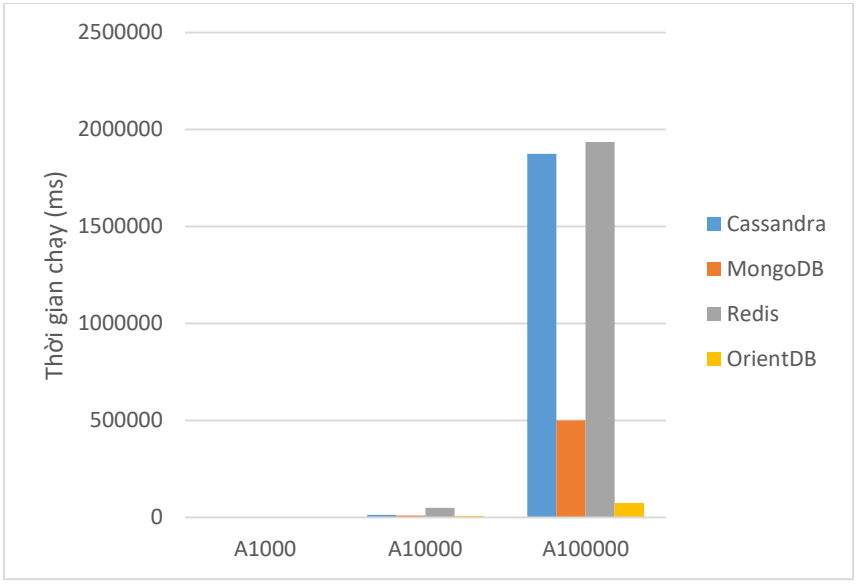
Redis thực hiện tỷ lệ Quét và Chèn kém nhất. Dựa trên dữ liệu hiện thị, tôi có thể kết luận rằng Redis không được khuyến nghị sử dụng trong trường hợp này. OrientDB là tốt nhất trong việc xử lý loại khối lượng công việc này như thể hiện rõ trong Hình 3.3.17 đến 3.3.22. MongoDB hoạt động tốt khi so sánh với những hệ quản trị cơ sở dữ liệu khác.



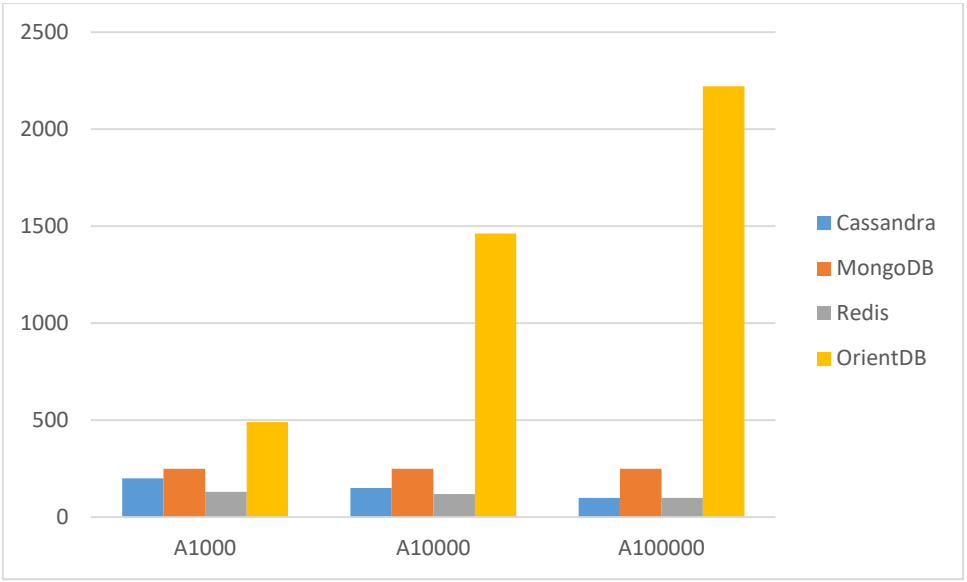
Hình 3.3.17 Kết quả thời gian chạy 10% quét - 90% chèn



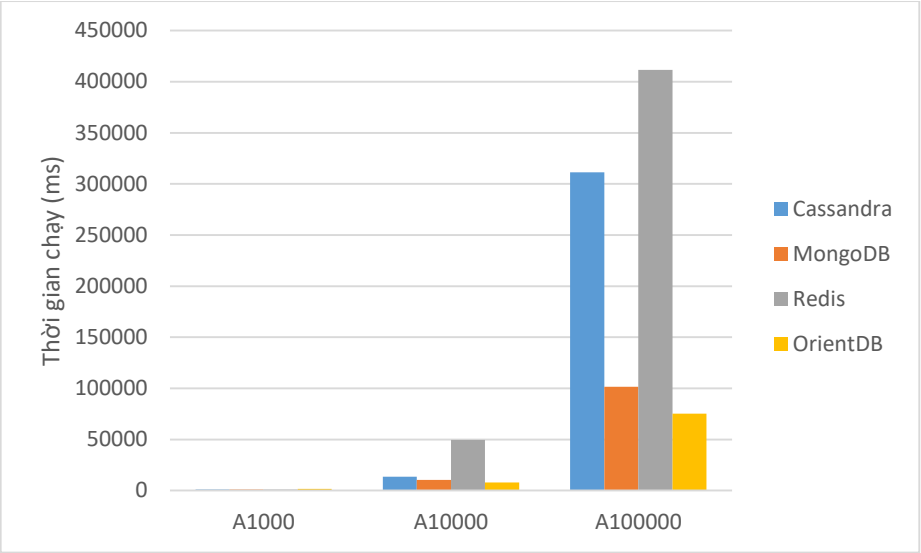
Hình 3.3.18 Kết quả thông lượng 10% quét 90% chèn



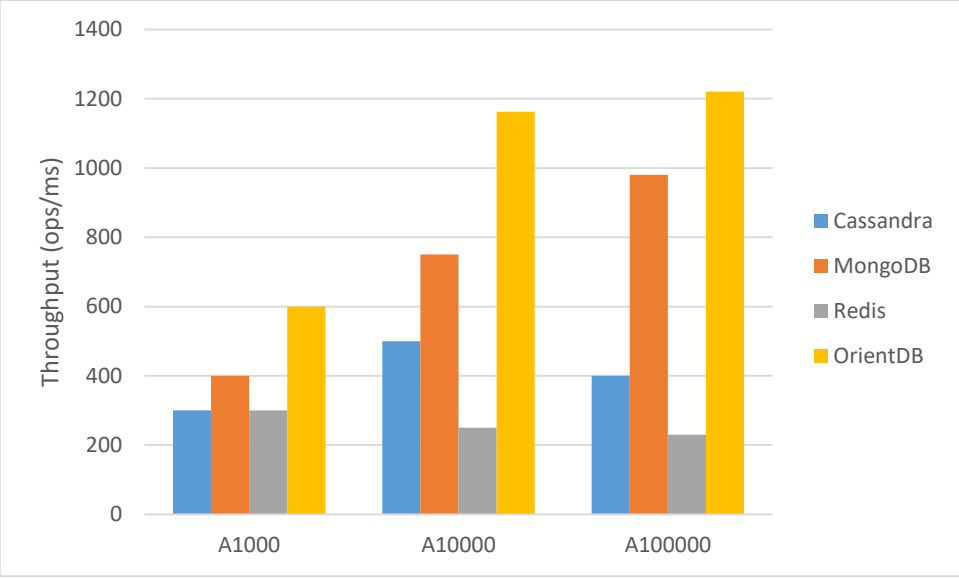
Hình 3.3.19 Kết quả thời gian chạy: 50% quét 50% chèn



Hình 3.3.20 Kết quả thông lượng 50% quét 50% chèn



Hình 3.3.21 Kết quả thời gian chạy: 10% quét 90% chèn



Hình 3.3.22 Kết quả thông lượng: 10%quét 90%chèn

Như vậy trong chương 3 tôi đã chạy các kịch bản để đo lường hiệu suất của các hệ quản trị cơ sở dữ liệu NoSQL và đưa ra các thông tin đánh giá hiệu suất cơ bản của từng hệ quản trị cơ sở dữ liệu.

Chương 4: Kết luận

Việc so sánh các hệ quản trị cơ sở dữ liệu NoSQL khác nhau mạng lại giá trị thực tiễn giúp làm rõ những ưu nhược điểm về giải pháp của từng công nghệ. Mục đích của nghiên cứu này là đo lường và so sánh hiệu suất của bốn hệ quản trị cơ sở dữ liệu: Redis, MongoDB, Cassandra, và OrientDB. Đây là các đại diện được chọn từ các họ cơ sở dữ liệu khác nhau. Dựa trên kết quả đánh giá các hệ thống cơ sở dữ liệu NoSQL phổ biến nhất.

Với tính năng lưu trữ tệp, MongoDB được sử dụng như một hệ thống tệp (GridFS) giúp cân bằng tải và sao chép dữ liệu trên nhiều máy tính để lưu trữ tệp. Trong đó, GridFS chia một tệp ra thành các phần hoặc các đoạn và lưu trữ thành những tài liệu riêng biệt. Bạn có thể truy cập GridFS bằng tiện ích Mongofiles hoặc plugin cho Nginx và Lighttpd. Truy vấn ad hoc là một trong những tính năng tốt nhất của chương trình. Nó hỗ trợ các trường, truy vấn phạm vi và tìm kiếm các biểu thức để trả về các trường tài liệu cụ thể bao gồm các hàm JavaScript do người dùng xác định hoặc các truy vấn này được cấu hình và trả về mẫu kết quả ngẫu nhiên có kích thước nhất định. Bên cạnh đó, các trường trong MongoDB có thể được dùng để lập các chỉ mục chính và các chỉ mục phụ.

Cassandra là hệ quản trị cơ sở dữ liệu sử dụng nhật ký để lưu trữ tất cả các thay đổi đã thực hiện, trong khi các bản ghi được lưu trữ trong bộ nhớ cho lần xả đĩa tiếp theo. Việc sử dụng các cơ chế này và sau đó ghi tuần tự vào đĩa làm giảm số lượng hoạt động của đĩa đặc trưng bởi tốc độ thấp so với tốc độ của bộ nhớ dễ bay hơi. Do đó, những hệ quản trị cơ sở dữ liệu này đặc biệt được tối ưu hóa để thực hiện cập nhật, trong khi việc đọc tốn nhiều thời gian hơn khi so sánh với hệ quản trị cơ sở dữ liệu trong bộ nhớ.

OrientDB có hỗ trợ cho các lược đồ đầy đủ, lược đồ ít và các mô hình lược đồ hỗn hợp. OrientDB sử dụng thuật toán lập chỉ mục mới có tên MVRB-Tree, xuất phát từ Cây Đỏ-Đen và từ Cây B +; điều này được báo cáo có lợi ích của việc có cả chèn nhanh và tra cứu nhanh.

Phân tích kết quả đo lường cho thấy thời gian xử lý và thông lượng của MongoDB có nhiều lợi thế so với các hệ quản cơ sở dữ liệu NoSQL khác. Tuy nhiên, các kết luận này chỉ áp dụng cho việc so sánh trực tiếp bốn hệ quản trị cơ sở dữ liệu được chọn và chỉ trong các trường hợp đã được mô tả trong nghiên cứu.

Các ứng dụng của một số hệ quản trị cơ sở dữ liệu NoSQL:

Stt	NoSQL	Loại	Ứng dụng
1	Redis	Key-Value Database	- làm cache cho ứng dụng - dùng để lưu thông tin trong sessions, profiles/preferences của user…
2	MongoDB	Document	- ứng dụng big data, e-commerce, CMS. - lưu log hoặc history…
3	Cassandra	Column-Family	- ứng dụng trong 1 số CMS và ứng dụng e-commerce…
4	OrientDB	Graph	- Ứng dụng trong các hệ thống mạng nơ ron, chuyển tiền bạc, mạng xã hội (tìm bạn bè), giới thiệu sản phẩm (dựa theo sở thích/lịch sử mua sắm của người dùng)..

Bảng 4.1: Các ứng dụng của một sộ hệ quản trị dữ liệu NoSQL.

Kết quả của luận văn này là sự phát triển của phương pháp luận để đo lường hiệu suất hệ quản trị cơ sở dữ liệu trong những kịch bản đại diện. Điều này mang lại lợi ích to lớn cho các nhà nghiên cứu chủ đề này. Kỹ thuật được nghiên cứu ở đây có thể được áp dụng cho bất kỳ hệ quản trị cơ sở dữ liệu nào và kết quả sẽ cung cấp các giá trị cho một trường hợp sử dụng nhất định. Điều này chắc chắn tiết kiệm thời gian nghiên cứu và cho phép so sánh các hệ quản trị cơ sở dữ liệu nhanh chóng và có ý nghĩa. Phương pháp này có thể có lợi khi xem xét các đặc điểm của các hệ quản trị cơ sở dữ liệu có thể ảnh hưởng đến hiệu suất. Các vấn đề này đã được xem xét ở trên và đã được đưa vào nghiên cứu trong luận văn này. Các nhà nghiên cứu chủ để này sẽ có lợi từ việc xem xét các phương pháp đo lường hiệu suất của một số hệ cơ sở dữ liệu đã được thực hiện trong chương 4. Danh sách các công cụ và kỹ thuật có thể quyết định cách tiếp cận tốt nhất để xác định hiệu suất của các hệ quản trị cơ sở dữ liệu trong bất kỳ nghiên cứu nào trong tương lai.

Kết quả tổng thể của luận văn này là đưa ra câu trả lời về việc so sánh bốn hệ quản trị cơ sở dữ liệu phổ biến nhất từ những hệ cơ sở dữ liệu phi quan hệ. Điều này chắc chắn mang lại giá trị cho các học viên trong lĩnh vực CNTT và đặc biệt là trong lĩnh vực kỹ thuật phần mềm. Kết quả của nghiên cứu này giúp đưa ra quyết định lựa chọn hệ quản trị cơ sở dữ liệu tốt nhất trong một dự án cho các chuyên gia thiết kế hệ thống, chủ sở hữu sản phẩm, chủ dự án và các chuyên gia kỹ thuật phần mềm khác. Thiết kế kiến trúc tổng quan của sản phẩm và thảo luận lựa chọn các hệ quản trị cơ sở dữ liệu trong các dự án dựa nhiều vào hiệu suất là một hoạt động quan trọng đối với các kỹ sư phần mềm. Việc lựa chọn các công nghệ, bao gồm cả hệ thống quản lý cơ sở dữ liệu, ảnh hưởng đến toàn bộ vòng đời của sản phẩm và do đó để chọn một hệ quản trị cơ sở dữ lệu tốt là một quyết định đầy thách thức. Các kết quả được trình bày trong luận văn này cung cấp dữ liệu hữu ích để đưa ra các quyết định này. Tuy nhiên, hiệu suất chỉ là một khía cạnh duy nhất ảnh hưởng đến sự lựa chọn cơ sở dữ liệu với chi phí mua, chi phí bảo trì, các dịch vụ bổ sung, tính ổn định hoặc quyền truy cập vào mã nguồn là ví dụ về các tiêu chí khác có thể ảnh hưởng đến việc lựa chọn. Việc mở rộng nghiên cứu với phân tích và so sánh hiệu suất của các công cụ cơ sở dữ liệu khác nhau có thể có khả năng làm tăng giá trị của nghiên cứu hơn nữa.

Cụ thể hơn, trong tương lai, tôi sẽ tiếp tục nghiên cứu các vấn đề sau: làm thế nào để cân bằng hiệu quả và chi phí trong lựa chọn NoSQL; so sánh NoSQL ở các khía cạnh khác, chẳng hạn như độ trễ hoạt động, hiệu quả mở rộng theo chiều ngang, v.v. Tôi cũng có kế hoạch thực hiện nghiên cứu về cách tối ưu hóa sau đó so sánh và chọn NoSQL trong một ứng dụng cụ thể.

Tài liệu tham khảo

[1] Abubakar, Y., T. Adeyi, and I.G. Auta, Performance evaluation of nosql systems using ycsb in a resource austere environment. Performance Evaluation, 2014. 7(8).

[2] Amazon Web Services. http://aws.amazon.com/.

[3] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J Abadi, David J DeWitt, Samuel Madden, and Michael Stonebraker. A comparison of approaches to large-scale data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pages 165–178. ACM, 2009.

[4] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In Proceedings of the 1st ACM symposium on Cloud computing, pages 143–154. ACM, 2010

[5] Carsten Binnig, Donald Kossmann, Tim Kraska, and Simon Loesing. How is the weather tomorrow?: towards a benchmark for the cloud. In Proceedings of the Second International Workshop on Testing Database Systems, page 9. ACM, 2009.

[6] Cooper, B., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R.: Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC '10). ACM, New York, NY, USA, 143-154, 2010.

[24] "Couchbase." Internet: http://www.couchbase.com.

[25] "Cassandra." Internet: http://cassandra.apache.org.

[7] David J DeWitt. The wisconsin benchmark: Past, present, and future., 1993.

Dede, E., et al., An Evaluation of Cassandra for Hadoop. IEEE CLOUD, 2013. 2013: p. 494-501.

[8] Dhruba Borthakur. Facebook has the world’s largest hadoop cluster! http://hadoopblog. blogspot.co.uk/2010/05/facebook-has-worlds-largest-hadoop.html, May 2010. Last Accessed: 2014.07.19.

[9] Eben Hewitt. Cassandra: the definitive guide. O’Reilly Media Inc., 2010

Erinle, B., Performance Testing with JMeter 2.9. 2013: Packt Publishing Ltd.

[10] Falko Bause. Queueing petri nets-a formalism for the combined qualitative and quantitative analysis of systems. In Petri Nets and Performance Models, 1993. Proceedings., 5th International Workshop on, pages 14–23. IEEE, 1993.

[11] Ghazal, A., et al. BigBench: towards an industry standard benchmark for big data analytics. in Proceedings of the 2013 ACM SIGMOD international conference on Management of data. 2013. ACM.

[12] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. Communications of the ACM, 51(1):107–113, 2008

[13] Michael Stonebraker and Rick Cattell. 10 rules for scalable performance in'simple operation'datastores. Communications of the ACM, 54(6):72–80, 2011

Pirzadeh, P., et al., Performance evaluation of range queries in key value stores. Journal of Grid Computing, 2012. 10(1): p. 109-132.

[23] "MongoDB." Internet: http://www.mongodb.org.

[14] Rackspace. http://www.rackspace.co.uk/.

[22] "Redis." Internet: http://redis.io.

[15] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. ACM SIGACT News, 33(2):51–59, 2002.

[16] Evans, D., The internet of things. How the Next Evolution of the Internet is Changing Everything, Whitepaper, Cisco Internet Business Solutions Group (IBSG), 2011. 1: p. 1-12.

[17] Tang, E., & Fan, Y. (2016). Performance Comparison between Five NoSQL Databases. 2016 7th International Conference on Cloud Computing and Big).

[18] Slideshare. HBase Vs Cassandra Vs MongoDB - Choosing the right NoSQL database. slideshare 2014 [cited 2016 20 Oct]; Available from: http://www.slideshare.net/EdurekaIN/no-sqldatabases-35591065.

[19] M. Löffler, Chui, M., and R. Roberts, The internet of things. McKinsey Quarterly, 2010. 2(2010): p. 1-9.

[20] TCP Benchmarks. http://www.tpc.org/information/about/history5.asp.

[26] Wang, L., et al. Bigdatabench: A big data benchmark suite from internet services. in 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA). 2014. IEEE.

[27] www.softwaretestinghelp.com/best-iot-examples/

[28] www.technologymag.net/hoa-ky-tai-tro-giai-phap-luoi-dien-thong-minh-cho-viet-nam/

[29] Yuri Gurevich Comparative Survey of NoSQL/ NewSQL DB Systems